Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Master's diploma thesis

in the field of study Data Science

Automatic Audio Chord Recognition

inż. Martyna Majchrzak

student record book number 298826

thesis supervisor

prof. dr hab. inż. Jacek Mańdziuk

WARSAW 2024

Abstract

Automatic audio chord recognition

This work explains the music theory needed to understand Audio Chord Recognition (ACR). It describes how it was done historically, with systems based first on templates, then Hidden Markov Models (HMM), and finally neural networks, including convolutional (CNN), recurrent (RNN), and Transformer-based architectures.

The purpose of this work is to compare two neural network models (BTC and HT) based on the Transformer architecture for recognizing a chord sequence in an audio recording and investigate the usefulness of an artificially generated dataset for this task. To achieve this, the models are trained on different combinations of the Artificial Audio Multitracks (AAM), Schubert's Winterreise Dataset and McGill Billboard Dataset and evaluated with three metrics: Root, MajMin and Chord Content Metric (CCM).

The experiments prove that even though there are certainly differences in complexity and structure, AAM can be useful in certain scenarios, for example, enriching a smaller training dataset of music composed by a human or even as a training set for a model which purpose is to predict chord sequences in pop music, if no other data is available.
Keywords: Automatic Chord Recognition, Neural Networks, Transformer, Artificial Audio Multitracks

Streszczenie

Automatyczne rozpoznawanie akordów w nagraniach audio

Niniejsza praca wyjaśnia teorię muzyki potrzebną do zrozumienia zadania jakim jest rozpoznawanie akordów w nagranich audio (ACR). Opisuje, w jaki sposób to zadanie realizowane było historycznie, z pomocą systemów opartych najpierw na szablonach, następnie na Ukrytych Modelach Markowa (HMM), a w końcu na sieciach neuronowych, w tym architekturach korzystających z sieci konwolucyjnych (CNN), rekurencyjnych (RNN) oraz Transformera.

Celem pracy jest porównanie dwóch modeli sieci neuronowych (BTC i HT) do rozpoznawania sekwencji akordów w nagraniach audio opartych na architekturze Transformer i zbadanie przydatności sztucznie wygenerowanego zestawu danych do tego zadania. Aby to osignąć, modele te zostały wytrenowane na różnych kombinacjach zbioru Artificial Audio Multitracks (AAM), Schubert's Winterreise Dataset i McGill Billboard Dataset i ocenione za pomocą trzech metryk: Root, MajMin i Chord Content Metric (CCM).

Przeprowadzone eksperymenty dowodzą, że pomimo różnic w złożoności i strukturze, zbiór AAM może być przydatny w niektórych przypadkach, na przykład do wzbogacenia mniejszego zbioru danych treningowych składającego się z muzyki skomponowanej przez człowieka, lub nawet jako cały zbiór treningowy modelu, którego celem jest przewidywanie sekwencji akordów w muzyce popularnej, jeśli żaden inny zbiór nie jest dostępny.
Slowa kluczowe: Automatyczne rozpoznawanie akordów, Sieci neuronowe, Transformer, Artificial Audio Multitracks

Contents

1. Introduction

Automatic Chord Recognition (ACR), often also referred to as Automatic Chord Estimation (ACE), is one of the central tasks of Music Information Retrieval (MIR). MIR is an interdisciplinary area of research, that involves fields such as musicology, signal processing, informatics and machine learning. Some of the other tasks of MIR are music synchronization, which aims to align different representations of music; music structure analysis, which is the identification of important structural elements or segments of the recording or tempo; and beat tracking, which is a study of extracting tempo-related information from audio recordings.

Automatic Chord Estimation is one of the subtasks in the annual MIREX (Music Information Retrieval Evaluation eXchange) competition, It has been ongoing from 2005 to 2021, although this task has first appeared on it in 2008. MIREX is an evaluation framework that determines current state-of-the-art systems in different MIR tasks and it has been standardizing datasets and formats used by researchers in this area for many years now.

First chord recognition systems were mostly knowledge-based systems. However, with the development of data-driven methods, a shift toward this approach has been observed. For this ACR research, neural network approaches have arguably been the most prominent, with architectures such as CNNs, RNNs, hybrid CRNNs, VAE and, more recently, Transformers being utilized.

Chord recognition in itself is a complex task, and obtaining reference data for it is challenging, as manual annotation is tedious and time-consuming, and even expert musicians might disagree on how a certain musical fragment should be labelled. Adequate datasets have historically been quite limited, but recent advances have expanded the possibilities.

The question of evaluating chord recognition systems results is not only a question of what metric to use, but also what set of labels (referred to as a vocabulary) and comparison strategy to choose. Due to those ambiguities, several approaches to this topic have emerged.

2. Related Works

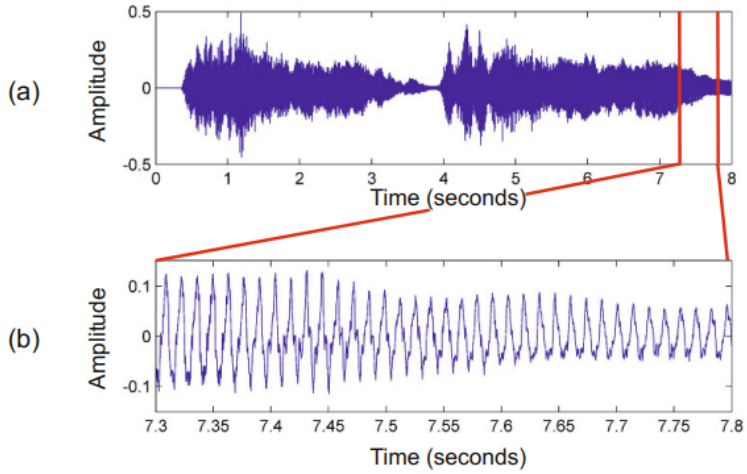
2.1. Theoretical underpinnings

This work will require an understanding of several concepts and tools from Music Theory and Music Information Retrieval. Most of those basics have been laid out in a great way in the *Fundamentals of Music Processing* by Meinard Müller [34]. Below I present the contents of this book that will be necessary later on, mainly based on chapter 1 - *Music Representations*, chapter 2 - *Fourier Analysis of Signals*, chapter 3.1 - *Audio Features* and chapter 5 - *Chord Recognition*.

2.1.1. Music Representation

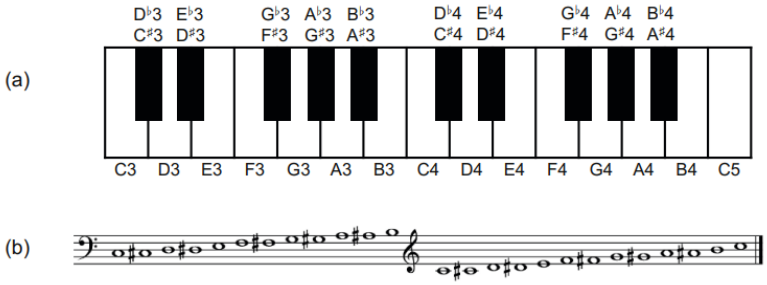
First, let us introduce some key musical concepts, starting with what a sound is. **Sounds** are acoustic waves transmitted through the air as pressure oscillations, generated by a vibrating object (f.e. human vocal cords, strings of an instrument). **Audio** is the transmission, and reception of sounds, that lie within the limits of human hearing. The changes in pressure can be represented on a pressure-time plot, called a **waveform**, which usually resembles a sinusoid - so it has a frequency, amplitude and phase. The higher the frequency of the sound, the higher it sounds to the human ear. An example of a waveform is presented in figure 2.1.

Figure 2.1: (a) Waveform of the first eight seconds of a recording of the first five measures of Beethovens Fifth (b) Enlargement of the section between 7.3 and 7.8 seconds. Source: [34]



The property of sound that allows a listener to distinguish the perceived height of a note is called its **pitch**. A **Musical tone** is a complex sound, so a mixture of different frequencies changing over time. Any of the sinusoids present in a musical tone is called a **partial**. Out of those partials the lowest one is called the **fundamental frequency**, and all the other ones - the **overtones**. The partials that are integer multiples of the fundamental frequency are called **harmonic partials**. As notes with fundamental frequencies in a ratio equal to any power of two (f.e. half, twice, four times) are perceived as very similar by the human ear, the concept of a **pitch class** was also created, covering those similar sounding notes. The interval between consecutive notes in such a series (so notes from which one has double of the others fundamental frequency) is called an **octave**. **Musical scale** is a discretisation of all possible pitches and even though there are many different ones, the most popular, and the one used in this work is a **twelve-tone equal tempered scale**, where an octave is divided equally into twelve scale steps. The difference between fundamental frequencies of two subsequent steps is called a **semitone**. Seven of the pitch classes are denoted by alphabet letters: C, D, E, F, G, A and B. The other pitch class names are created by adding either a sharp (#), which raises the note by a semitone, or a flat (b) which lowers it by a semitone. As the considered scale is a **chromatic scale**, so a scale ordered by pitch, we can use **Scientific Pitch Notation**, where a number denoting the **octave number** is added to the pitch class name. For example, pitch class A corresponding to the fundamental frequency of 220Hz is denoted by A3, 440Hz - A4, 880Hz - A5. This notation is presented on the piano and in Western music notation in figure 2.2.

Figure 2.2: (a) Section of piano keyboard with keys ranging from C3 to C5. (b) Corresponding notes using Western music notation. Source: [34]



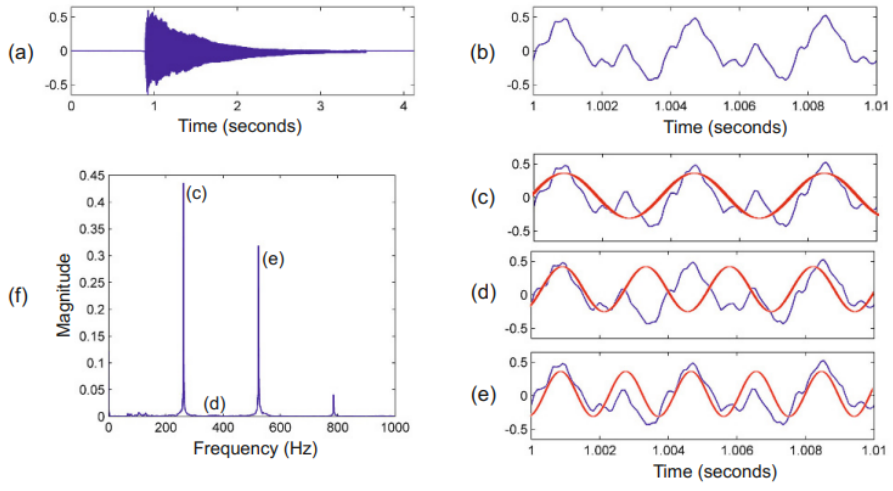
There are three fundamental ways of representing music: visual, symbolic and audio. The most prominent and well-known visual representation of music in Western culture is **sheet music**, which is a printed form of a musical score. It is usually written in the so-called **Western music notation**, which uses a five-line **staff**, a clef symbol and different musical symbols to represent the length of a note. The pitch of a given note is represented by the relative

position of the symbol on the staff. The symbolic representation of music is a machine-readable data format that explicitly represents musical entities. One commonly used protocol worth mentioning here is the **Musical Instrument Digital Interface (MIDI)**, where event messages specify pitches, velocities and other parameters to generate the intended sounds. The other symbolic music representation is refered to as **score representation**. It is defined to explicitly provide information about musical symbols such as the staff system, clefs, time signatures, notes, etc. One of the existing formats is **MusicXML**, which follows the general XML (Extensible Markup Language) paradigm. Finally, the audio representation is the explicit acoustic sound wave, commonly saved in a WAV or MP3 file format. It is important to remember that none of the above representations contain the full information about the particular piece of music. For example, while the score representations yield all the precise lengths and pitches of all the notes, they lack the qualities of a particular performance of a musical piece that is preserved in an audio representation.

2.1.2. Fourier Analysis

The **Fourier transform** is one of the most fundamental tools in signal processing. It transforms a signal representation dependent on time into a representation dependent on frequency. It breaks down a waveform into sinusoids of various frequencies and calculates the magnitude coefficients for those frequencies. The effects of applying a Fourier transform to a waveform of a note C4 is presented in figure 2.3.

Figure 2.3: (a) Waveform of a note C4 (261.6 Hz) played on a piano. (b) Zoom into a 10-ms section starting at time position t = 1 sec. (c-e) Comparison of the waveform with sinusoids of various frequencies ω. (f) Magnitude coefficients dω in dependence on the frequency ω.
Source: [34]



In a digital form, only a finite number of time-value pairs of a continuous analog signal can be processed. In practice, the original signal is discretized by taking equally spaced samples from the entire time domain. Fourier transform defined on such signal, that uses finite sums is known as a **discrete Fourier Transform (DFT)**.

In its standard form, the information is averaged over the entire time domain. However, there exists a local variant of it, called **Short-time Fourier transform (STFT)**, which yields the time-frequency representation of signal. The signal is divided into chunks with a so-called **window function**, which needs a specified **window size** and **hop size**. It is common to define a hop size smaller than the window size (f.e half its size), which results in the same parts of signal being analyzed multiple times in the subsequent steps.

A common way of visualizing the STFT is a **spectrogram**, which is a two-dimensional representation of the squared magnitude. As mentioned before, human perception of pitch is logarithmic in nature, and to emphasize musical or tonal relationships, the frequency axis is often plotted in a logarithmic fashion, resulting in a **Log-frequency spectrogram**.

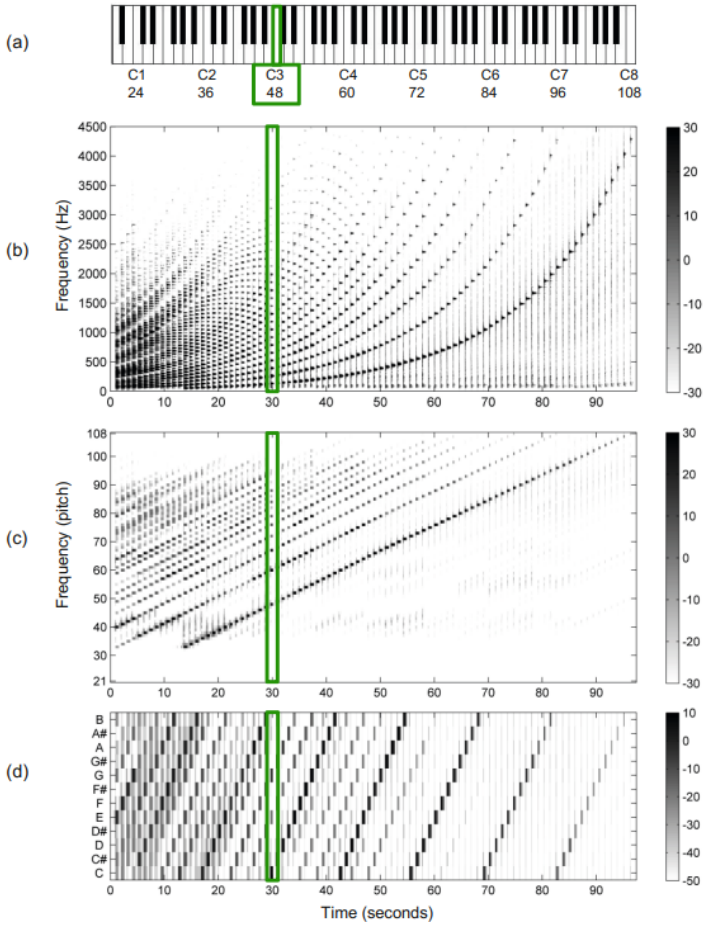
An alternative to STFT is the **Constant Q Transform (CQT)** [2] that uses a logarithmically spaced frequency axis and better reflects the perception of musical pitch and tonal relationships in human hearing. It is, however, also more computationally complex. It was first used for chord estimation in [35].

2.1.3. Chroma Features

Based on the log-frequency spectrogram, which is a time-frequency representation, one can define a **chromagram**. First, let us remind that every pitch can be separated by a chroma attribute that is an element of a 12-element set {C, C#, D, ... , B}, and an octave number. Each of the chroma attributes is associated with a set of frequencies of the subsequent pitches: C1, C2, C3, C4 and so on. The **chroma features** (also sometimes called Pitch Class Profiles or PCP) sum up all the coefficients of those frequencies from a log-frequency spectrogram into a single coefficient. Figure 2.4 visualizes the described representations of a recording on different notes on a piano - the regular spectrogram, log-frequency spectrogram and the chromagram.

Several techniques can be applied while transforming the time-frequency representation into one of the variants of the chroma representations, such as **smoothing**, **logarithmic compression**, **spectral whitening**, **normalization** or **quantization**. Logarithmic compression is an alternative to using the decibel scale. The compressed version of a spectrogram, log-frequency spectrogram or a chromagram is obtained by applying the compression function $\Gamma_\gamma(v) := log(1 + \gamma * v)$ on its values. The goal is to reduce the differences between very small

Figure 2.4: Various representations for a recording of the chromatic scale played on a real piano. The scale ranges from A0 (p = 21) to C8 (p = 108). (a) Piano keys representing the chromatic scale. (b) Magnitude spectrogram. (c) Pitch-based log-frequency spectrogram. (d) Chromagram. For visualization purposes the values are encoded by shades of gray using a logarithmic scale. The C3 (p = 48) played at time t = 30 sec is highlighted by rectangular frames. Source: [34]



and very large values in the representation. Normalisation is performed to make a chromagram invariant to changes in the dynamics of the music, and it is achieved by applying a suitable norm to the values of the representation.

In 2010 M.Mauch et al. introduced a method of calculating NNLS (Non-Negative Least Squares) Chroma features freques [31], that is commonly used in research to this day ([22], [37]).

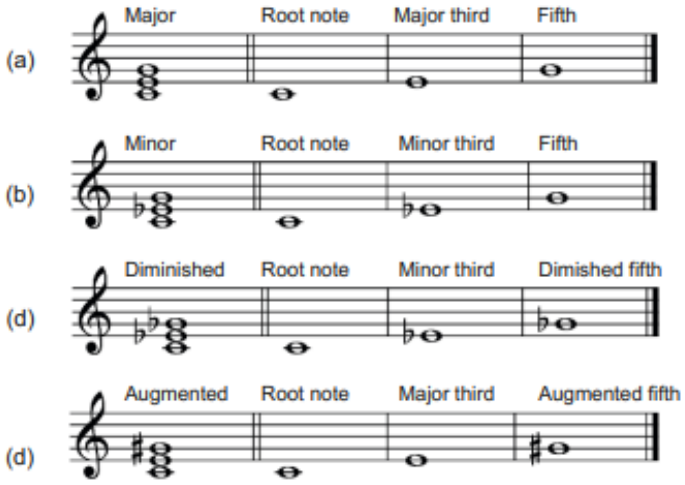
2.1.4. Chord Recognition

A **chord** is typically defined as a musical construct that consists of three or more notes. An **interval** is loosely defined as the difference between two pitches. We previously defined

an interval called octave (which is the distance between a pitch and another pitch with double its fundamental frequency) and a twelve-tone qual-tempered scale, where the octave is divided into 12 steps, with an interval of a **semitone** between each of them. Chord recognition is a classification task, which usually assumes the audio recording or calculated chroma features as a given, and takes the form of segmentating it into chunks and determining the chord label for each of them. The question of defining the set of those chord labels is, however, a complex one.

A triad is a chord that consists of three notes, the lowest of which is called a **root note**. A **major chord** is a note set where the first and second note differ by 4 semitones and the second and third by 3 semitones (1-4-3), and it tends to sound more cheerful to the human ear A **minor chord** is a note set where the first and second note differ by 3 semitones and second and third by 4 semitones (1-3-4), and it tend to sound more sad to the human ear. There are, however, many other common types of chords: for example **diminished chord** (1-3-3) and **augmented chord** (1-4-4). Those four types of chords are presented in figure 2.5.

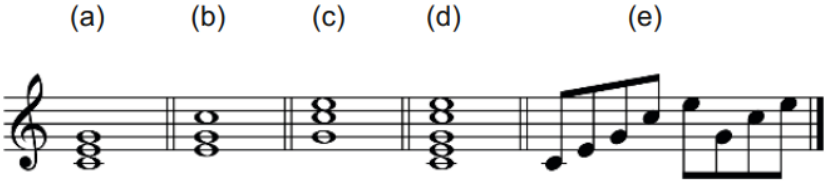
Figure 2.5: Various types of triads over the root note C4. (a) Major triad. (b) Minor triad. (c) Diminished triad. (d) Augmented triad. Source: [34]



The same chord can be played in many different, harmonically equivalent ways, which is presented in figure 2.6. When the chords lowest note is in the root, the chord is said to be in **root position**, when not, it is said to be **inverted**. **Octave doubling** is playing the notes that are already present in the chord, but an octave lower/higher. Finally, the notes of a chord do not necessarily need to be played all at the same time - when they are played one after another, resulting in a **broken chord**.

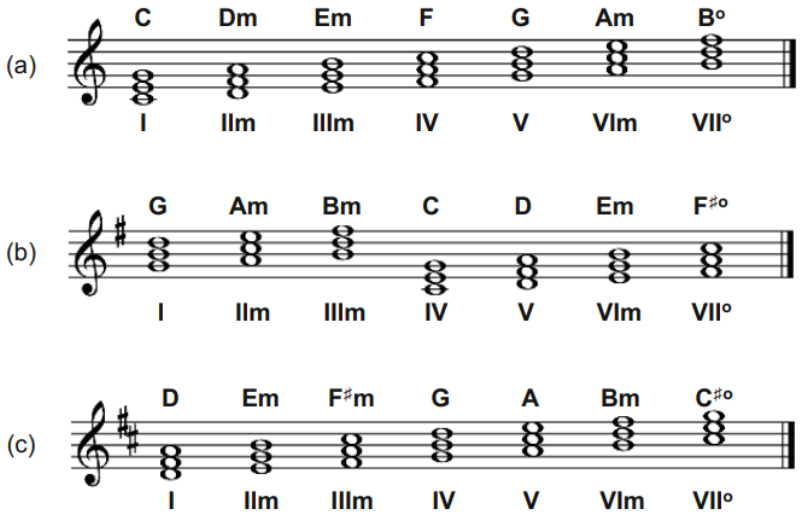
There is also another way of denoting the chords that is relative to the key note on the scale, using the Roman numerals (I, II, III, IV, ...), an example of which is presented in figure 2.7.

Figure 2.6: Variants of the C major chord. (a) Root position. (b) First inversion. (c) Second inversion. (d) Octave doubling. (e) Broken chord. Souce: [34]



In the case of minor chords, the suffix m is added to the numeral.

Figure 2.7: Roman numerals for the chords within a major scale. (a) C major scale. (b) G major scale. (c) D major scale. Source: [34]



It is important to note the concept of **chord progression**, which is an arrangement of chords over time. Its analysis can be quite helpful in chord recognition since some chords are much more likely to appear after a certain chord than others. Some of popular chord progression include I - IV - V (for example: C:maj, F:maj, G:maj) or I - V - VI - IV (for example: C:maj, G:maj, A:min, F:maj).

2.2. Chord Recognition Systems

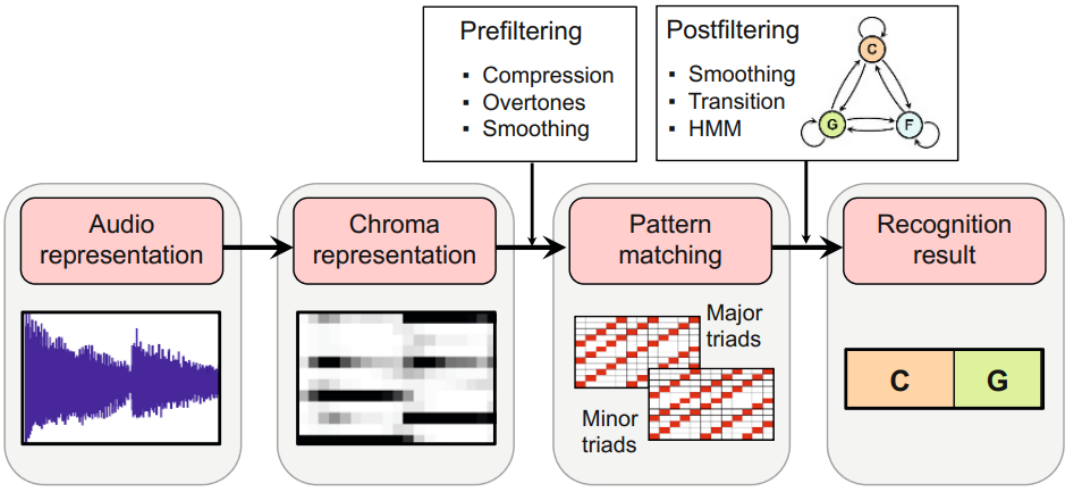
2.2.1. Template-Based Chord Recognition Systems

Up until a certain point, most chord recognition systems did not use machine learning techniques, but rather a pattern matching technique and heuristics - one of the first of such was a system proposed by T. Fujishima in [12], based on Pitch Class Profiles (PCP) created using

Discrete Fourier Transform.

Many common chord recognition systems follow the schema presented in figure 2.8. It consists of creating chroma features from the original audio recording, comparing the features with the predefined chord patterns and returning the label of a chord whose pattern maximizes a chosen similarity metric. Several enhancement techniques can be applied before or after the pattern-matching step. Those applied before are called **prefiltering** methods, and those applied after - **postfiltering**.

Figure 2.8: Overview of the components of a typical processing pipeline for automated chord recognition. Source: [34]



First, we will describe the most basic setup for this procedure. The common practice is to use the 25-element set which contains 12 Major chords, 12 Minor chords and 1 'non-chord' label N, so a set of possible labels is $\Lambda = \{C, C^{\#}, ..., B, Cm, C^{\#}m, ..., Bm, N\}$. First, the recording needs to be transformed into the sequence of feature vectors $X = (x_1, x_2, ..., x_n)$, where $x_n \in F, n \in [1 : N]$ and F is a feature space. For each of those chords, we need to create a **template** feature vector that will represent it in the feature space. Assuming the use of a twelve-step equal-tempered scale, chroma features can be represented as 12-dimensional vectors, where the values correspond to the set of pitch classes C, C#, D, ... B. As each chord can be defined as a subset of that set, the chord templates can simply be vectors $x = (x(0), x(1), ...x(11))^T$, where x(i)= 1 if the note corresponding to i is present in the chord. For example, in the case of the C major chord, the chroma vector is $t_C := x = (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0)^T$ There are many ways a similarity measure between such vectors can be defined and one of the simpler ones is the inner product of normalized vectors:

$$s(x, y) = \frac{\langle x|y\rangle}{\|x\| * \|y\|}$$

The label of a chord template that minimizes this measure is chosen as the chord label of the

piece of recording represented by a considered feature vector:

$$\lambda_n := \underset{\lambda \in \Lambda}{\operatorname{argmin}} s(t_\lambda, x_n)$$

The evaluation of the assigned labels λ_n is done against the **ground truth**, defined usually by manual annotation done by an expert. We will denote those reference labels as λ_n^{Ref}. A **true positive** (TP) is a correctly computed label ($\lambda_n = \lambda_n^{Ref}$). A **false positive** (FP) is an incorrectly computed label λ_n ($\lambda_n \neq \lambda_n^{Ref}$ and $\lambda_n^{Ref} \neq N$). A **false negative** (FN) is a reference label λ_n^{Ref} that does not agree with the computed label λ_n. The common measures are defined in the following way:

$$Precision = \frac{\#TP}{\#TP + \#FP}$$

$$Recall = \frac{\#TP}{\#TP + \#FN}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

There are some important ambiguities that one can encounter during the chord recognition task:

1. **chord ambiguities**, stemming from the fact that certain chords can have two or even three pitches in common

2. **acoustic ambiguities**, resulting from the superposition of harmonic overtones of all the notes being present in the chord, which can result in values in the chroma feature even higher than the values for the original notes.

3. different **tuning** (semitone up/down)

4. **segmentation ambiguities**, which can occur if the window frame or hop size in the STFT is not suitable for the musical piece at hand. This can be particularly important in the presence of broken chords.

Some of the **enhancement strategies** aiming at mitigating those problems are:

1. **Templates with harmonics**, which incorporate the energy of the harmonic overtones into the template, modeling the exponential decay of the energy by assuming the energy of a k^{th} partial is equal to α^{k-1} for some $\alpha \in [0, 1]$. For example, the first eight harmonics for a note with chroma C are

$$C, C, G, C, E, G, B^b, C$$

The template with harmonics for the chroma C would be:

$$t_C^h = (1 + \alpha + \alpha^3 + \alpha^7, 0, 0, 0, \alpha^4, 0, 0, \alpha^2 + \alpha^5, 0, 0, \alpha^6, 0)$$

A chord template with harmonics for the major chord C is obtained by summing up the templates over the chords chroma classes:

$$t_C^h = t_C^h + t_E^h + t_G^h$$

2. **Templates from examples**, which is an approach where one attempts to lean the templates from data - either by taking the average over feature vectors from all examples of more sophisticated supervised learning techniques.

3. **Spectral enhancement**, which uses the previously mentioned logarithmic compression to when creating chroma features.

4. **Prefiltering/temporal smoothing**, which applies an averaging filter onto the values of chroma features $x_n = (x_n(0), x_n(11))^T \in R^{12}, n \in [1 : N]$. The new, smoothed features are defined as:
$$x_n^L(i) := \frac{1}{L} \sum_{l=0}^{L-1} x_{n+l-\lfloor (L-1)/2 \rfloor}(i)$$
where L is the length of averaging filter (in frames).

The results of applying those enhancement techniques can be seen in figure 2.9

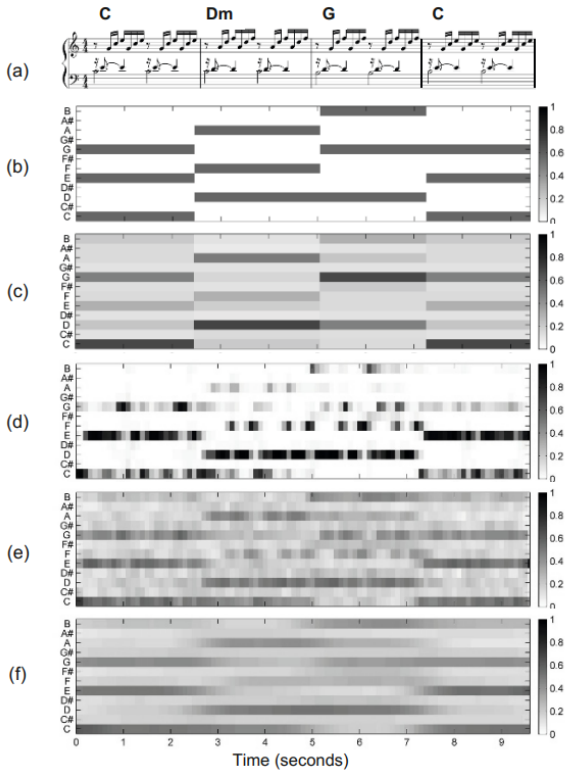
2.2.2. HMM-Based Chord Recognition Systems

While template-based chord recognition works strictly in a frame-wise fashion and each frame is classified independently from the others surrounding it, there is a way to incorporate the information from the neighbouring frames into the recognition system. The idea is to create a transition model that models the likelihood of passing over from one chord to another, and such a model is called **Hidden Markov Model** (HMM).

We assume that the observable features of a chord progression are generated by a **Markov chain** with chords as **states**, a **state-transition probability** matrix and **initial state probabilities**. This layer is 'hidden' as we cannot observe its parameters directly. Each observation has a defined **emission probability** of this type of observation being the output when in a certain state.

The task of chord recognition can be thought of as a task inverse of generating a sequence - uncovering the hidden state sequence that generates the given observation sequence with the highest probability. By utilizing algorithms like the **Viterbi algorithm**, the HMM estimates

Figure 2.9: Various types of enhanced chroma features. All chroma vectors are normalized with respect to the Euclidean norm and have a 10 Hz resolution. (a) Musical score and reference annotations. (b) Binary chord templates (corresponding to reference annotations). (c) Average chord templates obtained from training data (corresponding to reference annotations). (d) Original chroma features obtained from an audio recording. (e) Chroma features from (d) after applying logarithmic compression (using $\gamma = 10$). (f) Chroma features from (e) after applying smoothing (using L = 20). Source: [34]



the most likely sequence of chords given the observed audio features, essentially decoding the chord progression. The estimation of the model parameters is one to which there are no explicit solutions - procedures such as **Baum-Welch algorithm** are used to find the local maxima.

In practice, the 24 Major+Minor chords can be taken as the set of states. Since observation symbols (feature vectors) are 12-dimensional, real-values vectors and the HMM model as described above assumes a finite set of those symbols, the two possible solutions are the creation of a **codebook** from the observed features using clustering or application of a continuous HMM, with probability density functions instead of a discrete matrix of emission probabilities. There are also many options for defining the transition probabilities matrix - it can be defined manually by an expert based on their musical knowledge or learned from data. The most basic way would be to define it as the number of transitions from one chord to another, divided by all the occurrences of that first chord.

There are several extensions and variations of HMM, such as Dynamic Bayesian Networks (DBN) [30] or high-order HMMs [42].

One of the systems utilizing HMMs is a GMM/HMM chord estimation system [6], that combines them with Gausian Mixture Modelling. From beat-synchronized audio analysis, a multi-band chroma representation is generated, yielding four simultaneous feature representations. Each representation is individually patterned using its specific multivariate Gaussian Mixture Model (GMM). In this process, both chroma vectors and chord labels are adjusted to align with a C root. During the inference stage, the feature vector within the GMM is circularly shifted, allowing the system to derive four distinct likelihoods across all chord classes. This adjustment ensures the model's resilience against transposition variations. Subsequently, these four sets of chord class probabilities are collectively interpreted using a k-stream Hidden Markov Model (HMM). Through this joint interpretation, the system generates a single chord sequence synchronized with the beats, integrating information from multiple feature representations to achieve accurate chord estimation.

There has been some effort to systematize and compare the chord recognition systems created so far, such as in [40], that evaluates systems that were submitted to the MIREX competition in 2010, 2011 and 2012 or [33], that thoroughly describes and summarizes the research on the topic, including features, models, datasets, software and evaluation strategies. Cho and Bello in [7] explore the importance of individual system components, such as feature extraction, pattern-matching, pre-filtering and post-filtering, comparing techniques used for each of these steps and finding that the benefits of using complex chord models can be largely offset by an appropriate choice of features. It focuses primarily on the HMMs and Gaussian Mixture Modelling (GMM) based systems.

2.2.3. Chord Recognition Systems with Neural Networks

With the growing popularity of artificial neural networks, more and more systems utilizing them for the Chord Recognition tasks started to appear. There are two main ways that a neural network can be incorporated into the chord recognition pipeline:

- as a tool to create chroma features from audio recordings, such as [1], [53], [24], [25].
- as a tool that predicts the chord label based on the chroma features, such as [36], [17], [19] and [5].

One of the first works to use feed-forward neural networks for Chord Recognition was [36], which uses 12-valued Pitch Class Profiles and a simple network with 35 neurons in a hidden

layer, evaluated on a self-created database of chords (mostly guitar). [24] used a neural network comprised of rectifier units [13] to create chroma features, pairing them with a simple logistic regression classifier to match features to chords. Network with rectifier units was also utilized in a study [20] that strived for chord personalization using multiple reference annotators. More recent studies [21], [39] confirmed that annotator subjectivity is an important factor for chord recognition systems, but modern algorithms are powerful enough to tune themselves to the personal factors influencing particular annotators' decisions.

Convolutional Neural Networks

In [17] authors test several different architectures of Convolutional Neural Networks on five-second fragments of pitch spectra created using constant Q transform (CQT), achieving competitive results. Another work [53] compares the use of 'common' CNN and CNN bottleneck architecture, as well as SVM and HMM as the post-classifier and achieves over 90% WCSR. In [25], CNNs are used for extracting features that are later processed using Conditional Random Field (CRF) for chord label decoding. There has also been some effort to introduce musical knowledge into the chord representations, analyzing the distances between certain chords and proposing new training loss based on musical theory [4], improving the results of ACE systems based on CNNs.

CNNs have also been applied in studies on score-audio pairs: for learning Pitch-Class representations in the classical scenario [48] and for weakly aligned labels [46], with the use of Multi-label Connectionist Temporal Classification (MCTC) loss, initially proposed for speech.

Recurrent Neural Networks

Many chord recognition systems use two fundamental modules: an acoustic model which focuses on the discriminative aspect of the audio signal, and a language model that attempts to describe the temporal dependencies associated with the sequence of chord labels, such as harmonic progression. Recurrent Neural Networks are mainly used as the language model, being an alternative to Hidden Markov Models for frame-wise chord prediction from acoustic models. An example of utilizing Recurrent Neural Network is [1], which achieved state-of-the-art (at the time) results, about 93% Overlap Ratio on the major/minor task from MIREX competition dataset. It utilises Deep belief networks for feature extraction and RNNs as a 'language model', for inference utilizing an approach similar to the Viterbi algorithm used in HMMs, as well as beam search and dynamic programming. Another work that chose to replace HMMs with hybrid RNNs based on language models for modelling temporal dependencies between chords

is [44]. They used constant Q transform and a deep neural network for feature learning and proposed a modification to the beam search using a hash table, which reduces memory usage and run times. Two models were created: Acoustic Model (DNN) and Language model (RNN with LSTM units). The hybrid RNN model allowed to superimpose an RNN language model on the acoustic model predictions, improving them significantly. Research from 2017 [16] shows, that encoder-decoder-based LSTMs can do an even better job at modelling the chord progression from chroma features than simpler LSTM networks. However, the sensibility of using RNNs instead of HMMs has been challanged by Korzeniowski and Widmer in 2017 [26]. In later works [27] [23] the same authors explore the fact that, if applied on the level of chord sequences, RNNs can become powerful chord predictors and propose to replace the standard temporal model with a harmonic language model and a chord duration model that connects the chord-level predictions of the language mode; to the frame-level predictions of the acoustic model. For both of these they test different combinations of vanilla RNN, GRU and LSTM recurrent units.

While most of those studies use the 25 classes (N+major+minor) of chords, there has been some effort to create models more suitable for tasks with a much richer chord vocabulary ([32], [10], [8], [9]) facing the problem of extreme rarity of some of the classes. Y. Wu and W. Li in [52] propose a system based on Deep Residual Network (DRN) for feature extraction, and Bidirectional LSTM and Conditional Random Field (CRF) for classification and decoding, that is also suitable for larger vocabulary chord recognition. Later on, the same authors proposed a similar system [51] with a CNN-based feature extractor and BLSTM-CRF (Bidirectional Long Short Term Memory - Conditional Random Field) decoding model.

A question arises as to whether a data-driven approach and more complex architectures have the potential to significantly improve state-of-the-art methods. Humphrey and Bello conducted a comparative study [18] and suggested some insights into the field and possible remedies.

Convolutional Recurrent Neural Networks

Another architecture used for audio signal processing that became popular around 2017 is a hybrid approach - a combination of multiple layers of Convolutional Neural Network, followed by a Recurrent layer at the end. Systems based on this approach have been submitted to the MIREX 2018 [19] and MIREX 2019 [29]. This architecture, paired with HMM, has also been used for training with non-aligned annotations [50].

Variational Autoencoder

Recent works [49] also include statistical method that trains a neural chord estimator in a semi-supervised manner by constructing a Variational Autoencoder (VAE) with latent chord labels and features.

Transformer

With the popularization of the Transformer architecture [45], originally developed for tasks in Natural Language Processing, some researchers proposed applications for chord recognition. A Harmony Transformer [5] assigned the chord segmentation task to the encoder and the chord recognition task to the decoder. Another work introducing the BTC (Bi-directional Transformer for Chord recognition) [38] highlights the usefulness of the attention mechanism and visualizes the way the model works through attention maps. One of the advantages of the transformer architecture is the fact that there is no need for additional decoders such as HMMs or Conditional Random Fields (CRFs), so only one training phase is required.

2.3. Datasets

As creating the 'ground truth' annotations for music is a tedious and challenging task in itself, there has never been an abundance of available datasets for evaluating chord recognition systems. The most common problems with publishing the original audio recordings are copyright issues. There are, however, a couple of popular datasets used in other research papers that could be useful for this task. We will consider four in depth: three created from real audio recordings and one artificially generated.

2.3.1. The Beatles

The first common dataset made available to researchers was released by Chris Harte and collaborators in 2005 [15]. The Beatles collection is a dataset created from a corpus of twelve studio albums by the famous English rock band. It covers 180 songs, totaling 8 hours, 8 minutes, and 53 seconds of raw audio data. Even though those recordings are not included in the dataset itself, they are relatively easily found on streaming platforms online. A detailed description of the chord syntax, transcription process, and verification process for the Beatles collection can be found in Christopher Harte's PhD thesis [14] published in 2010.

The annotations are available as .lab files, a format made popular by a Music Information

Retrieval competition called MIREX [1]. They contain start time, end time and chord label - an example of which can be found in table 2.1. The N symbol is a so-called 'no chord' label, used for example for the silence at the beginning of a recording.

start-time	**end-time**	**label**
0.000000	2.612267	N
2.612267	11.459070	E
11.459070	12.921927	A
12.921927	17.443474	E
17.443474	20.410362	B
20.410362	21.908049	E
21.908049	23.370907	E:7/3

Table 2.1: Beginning of an annotation file for the song 'I Saw Her Standing There' from 'Please Please Me' album

There is only one chord notation, which is a very precise one - no simplifications or reductions seem to have been done. The authors identify chords including from one to six notes (triads being the most common one, covering almost 80% of the individual chord labels). This covers a massive vocabulary of 133 unique chord types (a chord type being, for example, X:maj, X:min, X:7, X:maj/5, where X is a pitch symbol). Apart from the chord annotations, there are additional keys and beats labels available.

This dataset has been made available as one of the Isophonics Datasets [2] (other ones include musical pieces by Zweieck, Queen and Carole King). It has been canonically and extensively used in a variety of different research papers and books, such as [34], [17], [1], [44], [53], [18], [24], [25], [10], [28], [19], [29], [50] and [38].

2.3.2. The McGill Billboard Project

The Billboard annotations dataset was first introduced in 2011 [3]. The goal of the project was to create a dataset with a broader range of artists and musical genres. To achieve that, the songs were sampled from the *Billboard* "Hot 100" weekly compilation of the most popular music in the USA. The audio tracks are not included due to copyrights. However, the authors were able to include audio features: NNLS chroma vectors [31] and tuning estimates from the

[1] https://www.music-ir.org/mirex/wiki/MIREX_HOME

[2] http://isophonics.net/content/reference-annotations-beatles

Chordino VAMP plugin[3]. which are helpful for chord recognition. As per the information on the website, it is also possible to request other types of features from the authors.

The file 'billboard-2.0-index.csv' files contain the entries for 1300 sampled songs, but only 890 out of those have the title, chroma features and annotations available. This file also contains additional metadata such as the artist, date when the song appeared on the chart and chart rank. Example contents of a chroma features file ('bothchroma.csv') is shown in table 2.2.

start-time	**v1**	**v2**	**...**	**v23**	**v24**
0.000000000	0.198482	0	...	0.562561	0.864485
0.046439909	0.310882	0	...	0.627636	0.904673
0.092879818	0.404969	0	...	0.686886	0.906338

Table 2.2: Beginning of an example chroma features file for the song 'I Don't Mind' by James Brown

The original, complete annotations include the chords, song structure, instrumentation and timing in a format resembling musical scores. However, the authors also include the MIREX style .lab files, with start time, end time and chord labels. One can download files with two different chord-label dictionaries - a more extensive and simplified one. Example contents of the annotation file with the broader dictionary for the Billboard dataset are shown in table 2.3. The N symbol is again a 'no chord' label.

start-time	**end-time**	**label**
0.0	7.3469387e-2	N
7.3469387e-2	1.51356009	N
1.51356009	1.8015782305999999	N
1.8015782305999999	3.529687074200001	A:min
3.529687074200001	5.257795917800002	A:min
5.257795917800002	6.985904761400003	C:maj
6.985904761400003	8.714013605000009	C:maj

Table 2.3: Beginning of an annotation file for the song 'I Don't Mind' by James Brown

This dataset is available online[4].

It has been used by multiple previously discussed studies, such as [1], [44], [18], [28] [5], [19],

[3]http://www.isophonics.net/nnls-chroma

[4]https://ddmal.music.mcgill.ca/research/The_McGill_Billboard_Project_(Chord_Analysis_Dataset)/

[29] and [50].

2.3.3. Schubert Winterreise

The Schubert Winterreise dataset is a multimodal dataset introduced in 2021[47]. It consists of Franz Schuberts 24-song cycle called 'Winterreise', composed in 1827 for voice and piano. One of its unique features is the fact that it comes with the original music in many different representations: audio .wav files, lyrics (in German) and scores in different formats (such as midi and pdf) are all included. Additionally, it contains both chord and note annotations for all of the songs (both the recording of nine different performances and the scores), with start and end times (in seconds) available. Out of those nine performances, only two of the actual recordings are included in the actual audio form due to copyright issues. The length of the raw audio data is 2 hours, 14 minutes and 16 seconds (1:07:31 for one of the performances and 1:06:45 for the other). Each chord label is given in 4 notations:

- shorthand notation
- extended notation with explicit intervals
- reduced to major and minor triads
- reduced to major and minor triads with bass note

The example of the contents of an annotation file for one of the songs is presented in table 2.4.

start	**end**	**shorthand**	**extended**	**majmin**	**majmin_inv**
0.46	4.58	C:min	C:(b3,5)	C:min	C:min
4.58	5.2	B:dim7/C	B:(b3,b5,bb7)/C	B:min	B:min/C
5.2	6.9	C:min	C:(b3,5)	C:min	C:min

Table 2.4: Beginning of an annotation file for the song no. 1 in the Winterreise cycle, entitled "Gute Nacht"

Moreover, the dataset contains annotations for the audio structure (repetitions of parts of the song), global and local keys, as well as additional metadata and files used for the preprocessing of the recordings.

The dataset is published under a Creative Commons Attribution 3.0 Unported license. It has been made available online[5]. It is a relatively new dataset, but it has already proved to be

[5]https://zenodo.org/records/5139893.YWRcktpBxaQ

quite useful for a variety of different studies, including local key estimation [43] and learning pitch-class representations [48] [46].

2.3.4. Artifcial Audio Multitracks

Artificial Audio Multitracks [37] is a state-of-the-art dataset of 3000 artificial music tracks with rich annotations based on real instrument samples and generated by algorithmic composition with respect to music theory, released in 2023. It is intended to be helpful for various music information retrieval tasks like music segmentation, instrument recognition, source separation, onset detection, key and chord recognition. As the audio is perfectly aligned to the original MIDIs, all annotations (onsets, pitches, instruments, keys, tempos, chords, beats, and segment boundaries) are absolutely precise. Furthermore, the authors conducted experiments that proved that this dataset is useful for neural network models for music segmentation, instrument recognition, and onset detection. However, no such study has yet been done for chord recognition.

For each track MIDI, mp3 files (as separate instruments and mixes) and annotations are available. The chord annotations are present in *XXXX_beatinfo.arff* files that follow the format shown in table 2.5.

start time in seconds	**bar count**	**quarter count**	**chord name**
0.0	1	1	'Fmaj'
0.65217394	1	2	'Fmaj'
1.3043479	1	3	'Fmaj'
1.9565217	1	4	'Fmaj'
2.6086957	2	1	'A#maj'

Table 2.5: Beginning of an annotation file 0001_beatinfo.arff in AAM dataset

This dataset is available under this link [6].

2.4. Metrics

The question of evaluating chord recognition systems results is not only a question of what metric to use, but also what set of labels (referred to as a vocabulary) and comparison strategy to choose. The metrics can be defined on two levels:

[6] https://zenodo.org/records/5794629

- single chord level - refers to a metric that compares two chords
- chord sequence level - refers to a metric that compares two sequences of chords, containing information about the durations of each labeled sequence piece

2.4.1. Single chord level

Root

This metric only compares the root of the two chords. It takes the value of 1 if they are the same and 0 if not. For example, C:maj and C:min chords have the same root note, so root(C:maj, C:min)=1.

MajMin

Only compares major, minor, and no chord labels. If the vocabulary contains more complicated chords, they are simplified and to one of the 25 classes and then compared. It takes the value of 1 if they are the same and 0 if not.

Mirex

An estimated chord is considered correct if it has at least three pitch classes in common with the ground truth annotation.

Chord Content Metric (CCM)

In 2022 J. Devaney proposed a novel chord estimation metric [11] that strives to take the (possibly overlapping) notes that the predicted and reference chords have in common. Unlike the previously defined metrics, it makes other values between 0 and 1. Let C be the number of predicted notes $\hat{y}$ in the ground truth correctly identified y and I be the number of insertions (extra predicted notes) in the estimated chord that are not present in the ground truth.

$$C = |y \cap \hat{y}|$$

$$I|\hat{y}\ y|$$

The accuracy measure is defined as:

$$A = \frac{C - I + |y|}{2|y|}$$

Metric implementation can be found on a Github repository[7].

[7] https://github.com/jcdevaney/chordEstimationMetric

2.4.2. Chord sequence level

WCSR \WAOR

The most popular metric was proposed for MIREX 13: the **Weighted Chord Symbol Recall (WCSR)**, often also called **Weighted Average Overlap Ratio (WAOR)**. It has been used by the majority of discussed studies ([53], [24], [25], [10], [28], [5], [19], [29] and [38]). WCSR is defined in [53] as the total duration of segments with a correct prediction:

$$WCSR = \frac{1}{N}\sum_{k=1}^{n} C_k$$

in which n is the number of test samples (songs), N is the total number of frames in all test samples, and C_k is the number of frames that are correctly detected in the kth sample. To determine which frames were correctly detected, a metric on the single chord level, such as Root or MajMin, should be used.

The metric is implemented in the `mir_eval` Python library [41], which offers metrics for measuring the performance of different MIR algorithms, including chord estimation.

Weighted Accuracy

Weighted Accuracy is a generalization of WCSR, more suitable for metrics that take values other than 0 and 1, such as CCM. It computes the desired metric over all chords in the sequence and then takes a mean weighted by the duration of each segment.

3. Methodology

This section describes the experimental setup, chosen model architectures, and data preprocessing steps. As mentioned in section 2.3.4 the authors of AAM claim that it is suitable for a plethora of MIR tasks and present experiments that test its usability for some of them. However, there are no experiments investigating it's usefulness for chord recognition. The aim of this work is to test it by training neural network model architectures on it and compare the results to models trained on real music. Training sets with a combination of different datasets are also considered.

3.1. Model architectures

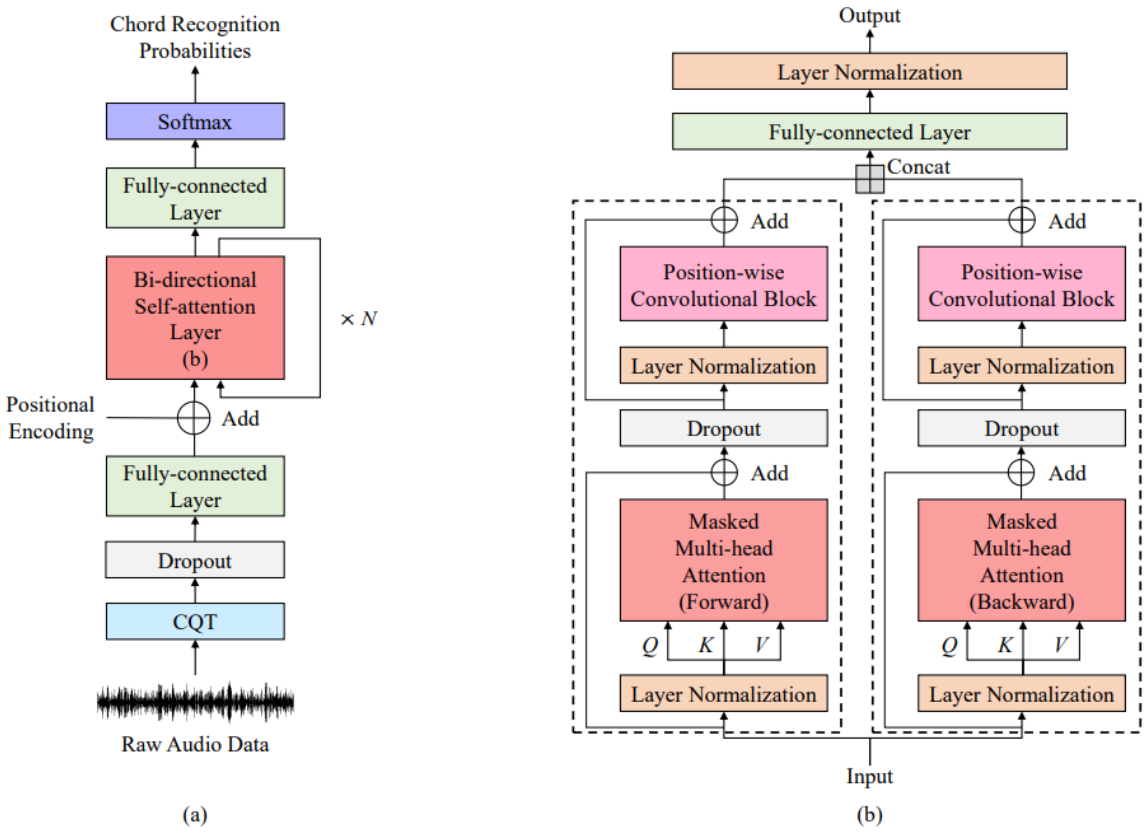
For the experiments, two models were used, both based on the transformer architecture: the Bi-directional Transformer for chord recognition (BTC) [38] and the Harmony Transformer (HT) [5]. They were chosen mostly due to being proposed fairly recently (2019) and the availability of the implementation code. They both incorporate the information from the surrounding frames into the prediction process, so they predict a sequence of chords instead of a single chord.

3.1.1. BTC

The BTC architecture is shown in figure 3.1. The process begins with extracting CQT features from the input audio, followed by a dropout layer, a fully connected layer, and positional encoding. This is succeeded by N bi-directional self-attention layers and a final fully connected layer. The self-attention mechanism determines the degree of focus on the value of a key time frame when predicting the chord at a query time frame. The bi-directional masking technique is employed to retain information by preventing the attention mechanism from processing the entire input at once and allowing the model to utilize context from both before and after the target time frame. Each self-attention layer includes a masked multi-head attention block (as in the original Transformer decoder) and a Position-wise Convolutional Block which consists of a 1D convolutional layer, a ReLU activation function, and a dropout layer. This block captures

the sequence order information and leverages features from neighboring time frames. During training, the negative log-likelihood is used as the loss function.

Figure 3.1: Structure of BTC. (a) shows the overall network architecture and (b) describes the bi-directional self-attention layer in detail. Dotted boxes indicate self-attention blocks. Source: [38]



For the purpose of this study, the hyperparameters proposed by the authors of BTC were used. They are presented in table 3.1.

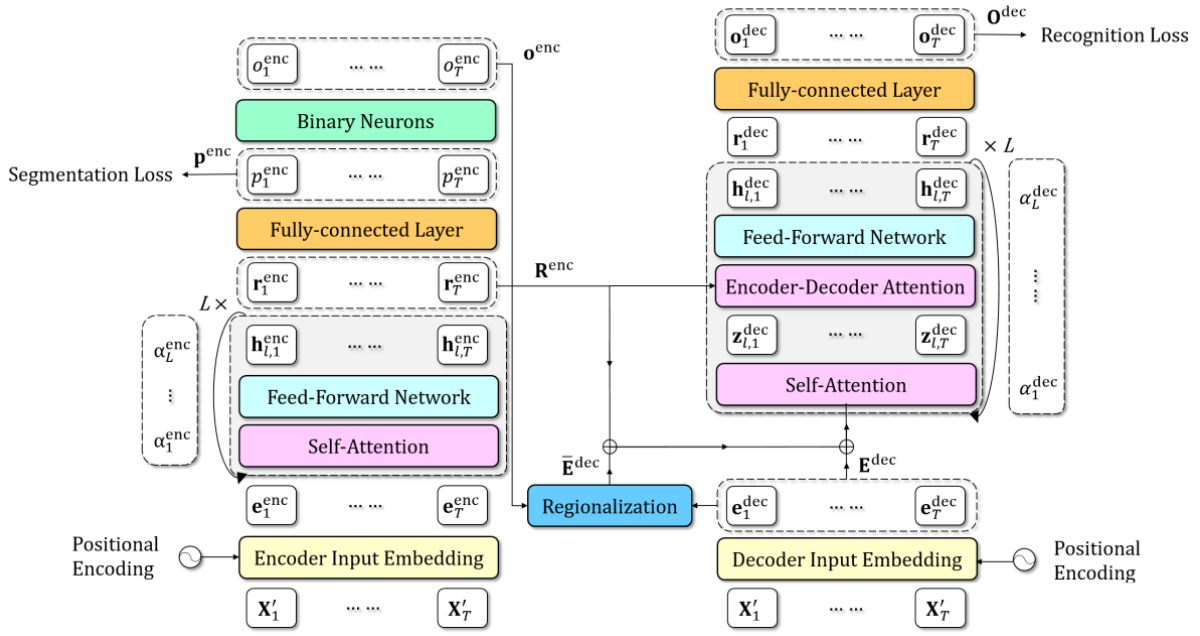
Bi-directional self-attention layer	layer repetition (N)	8
	self-attention heads (n_h)	4
	dimensions of Q, K, V and all the hidden layers	128
Position-wise convolutional block	block repetition (n_C)	2
	kernel size	3
	stride	1
	padding size	1

Table 3.1: Hyperparameters of BTC. Source: [5]

3.1.2. HT

The HT architecture, shown in figure 3.2, uses the encoder-decoder architecture of the Transformer. It can be used to recognize chords from both audio and symbolic music representation but takes precalculated features (such as chroma features) as input. The input embedding (for both the encoder and the decoder) is a mapping of a segment from the input data through a multi-head attention and a feed-forward network, with absolute positional encoding added. The encoder performs the task of chord segmentation on the data sequence. It is composed of L layers, each containing a multi-head self-attention and a feed-forward unit. The output of the encoder is a binary sequence o^{enc}, where 1 indicates a point of chord change and 0 is the value for all other points. The decoder performs the actual chord recognition task on the input data sequence, combined with the segmentation results from the encoder. It also consists of L layers, each of which has an additional encoder-decoder attention module. Binary cross entropy is used for the segmentation loss and categorical cross entropy for the chord recognition loss. The total loss function in the Harmony Transformer is a weighted sum of the two above, balanced by coefficients λ_1 and λ_2 that can be configured.

Figure 3.2: Structure of the Harmony Transformer. During training, the chord change prediction p^{enc} from the encoder is used to calculate the segmentation loss, while the decoder output O^{dec} is used to calculate the recognition loss. Source: [5]



For the purpose of this study, the hyperparameters proposed by the authors of HT were used. They are presented in table 3.2.

Multihead attention	layer repetition (L)	2
	self-attention heads	8
	embedding size	512
loss balancing coefficients	λ_1	3
	λ_2	1

Table 3.2: Hyperparameters of HT

3.2. Datasets

The BTC model was trained on a subset of 221 songs from Isophonics dataset [1] and 185 songs from UsPop2002 [2]. Due to copyright issues, the datasets do not include audio files, only annotations. The authors of BTC collected the music themselves from online music service providers and also couldn't include them when sharing the training code. They did, however, include a final model trained on this data. Because this architecture is based on raw audio, the only datasets that could be used to train and evaluate it were the artificially created AAM dataset and a small classical music Winterreise dataset.

The HT model was trained on the Billboard dataset, and its NNLS chroma features and annotations are included in the code repository. The experiments with this model were conducted using Billboard, AAM and Winterreise. The NNLS chroma features were calculated on the AAM and Winterreise audio files.

3.2.1. Exploratory data analysis

This section is a short analysis of the songs in the chosen datasets. Table 3.3 shows mean and standard deviation of long length in each dataset. AAM songs are on average the shortest and have the smallest standard deviation.

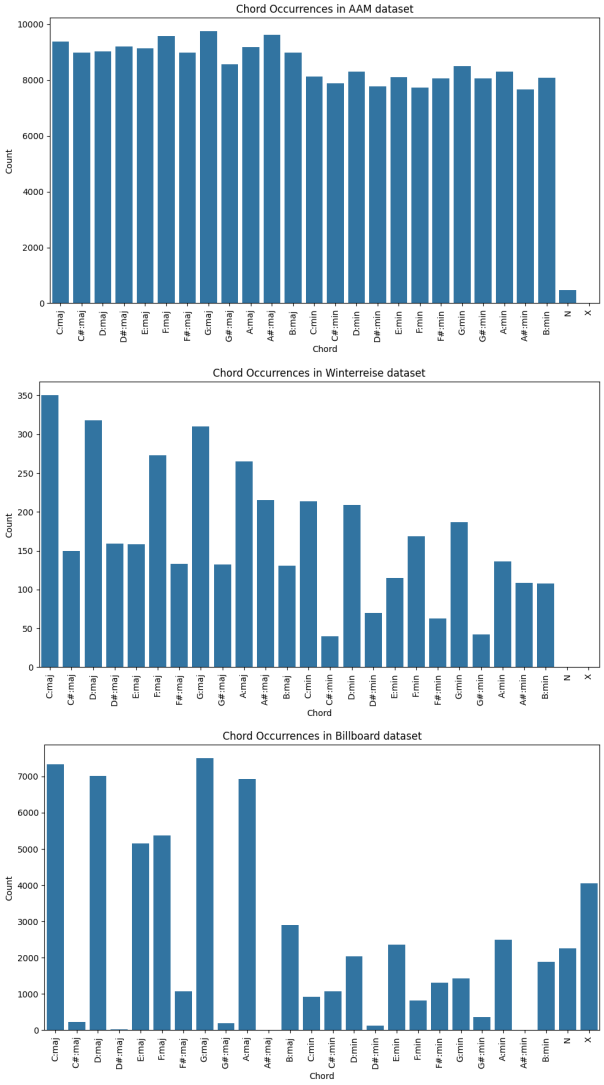
dataset	mean song length [sec]	std of song length [sec]
Winterreise	166.67	67.64
AAM	156.55	17.73
Billboard	216.36	66.62

Table 3.3: Mean and standard deviation of song length in each dataset

[1] http://isophonics.net/datasets

[2] https://github.com/tmc323/Chord-Annotations

Figure 3.3: Number of occurrences of each chord from the vocabulary in the AAM, Winterreise and Billboard dataset. If the chord label is in the annotation files for several consecutive rows, it is counted as one occurrence.



It is clearly visible in figure 3.3 that the distribution of chords in the AAM dataset is much more uniform than in the datasets that consist of real songs. In the Billboard dataset some chords, such as A#:maj and A#:min are never present. The difference in the number of Non-chord labels between AAM and Winterreise versus Billboard, is because annotation files for the first two start at the moment where a chord is present and recognized, and for the latter, the annotations start at second 0.00 and the silence at the beginning and end of the file is annotated as 'N'. A couple of the Non-chord labels in AAM tracks are the rows from the annotation files where a value 'BASS NOTE EXCEPTION' was found in the original dataset and converted into the 'N' label.

Figures 3.4, 3.5 and 3.6 show counts of each possible chord progression in each of the datasets.

Again, the patterns of chord changes in AAM seem much more uniform and consistent. However, in all dataset there is a clear linear pattern that represents the most likely distance between the currently played chord and the next one. This confirms the existence of popular chord progressions mentioned in section 2.1.4, such as [C:maj, F:maj, G:maj] or [C:maj, G:maj, A:min, F:maj].

Figure 3.4: Number of changes from one chord to another in the AAM dataset. Only changes to a different chord are counted.

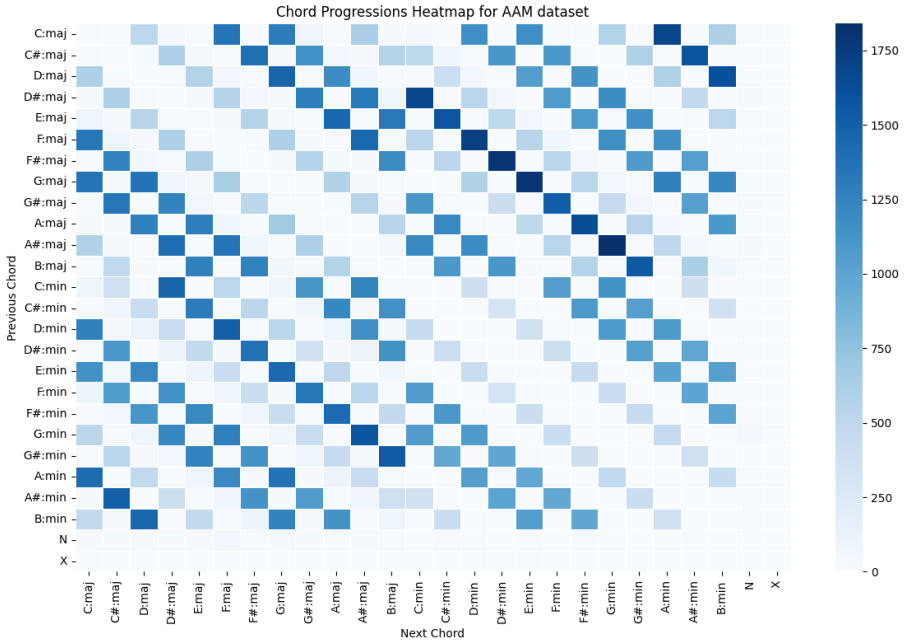


Figure 3.5: Number of changes from one chord to another in the Winterreise dataset. Only changes to a different chord are counted.

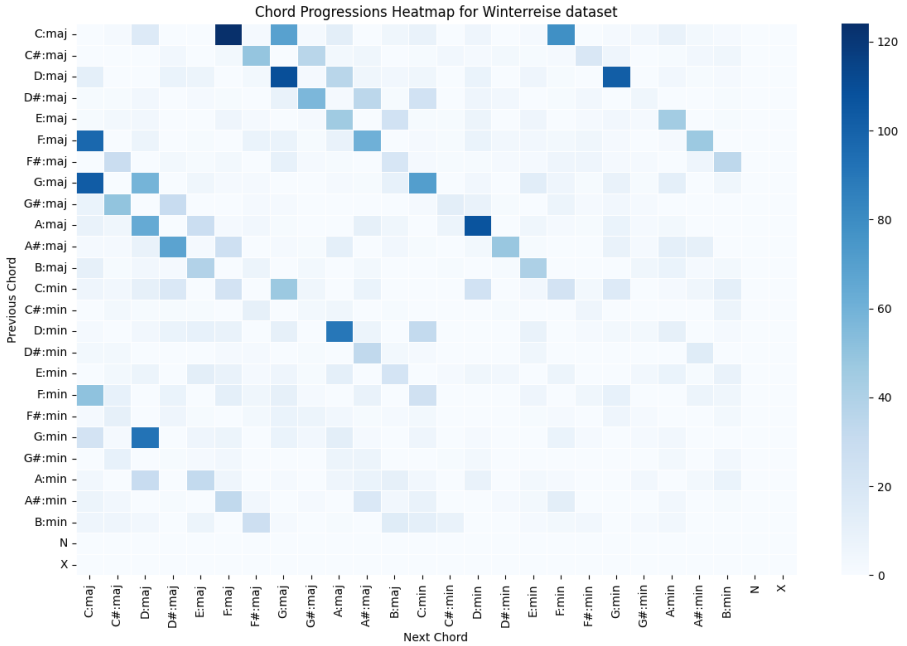
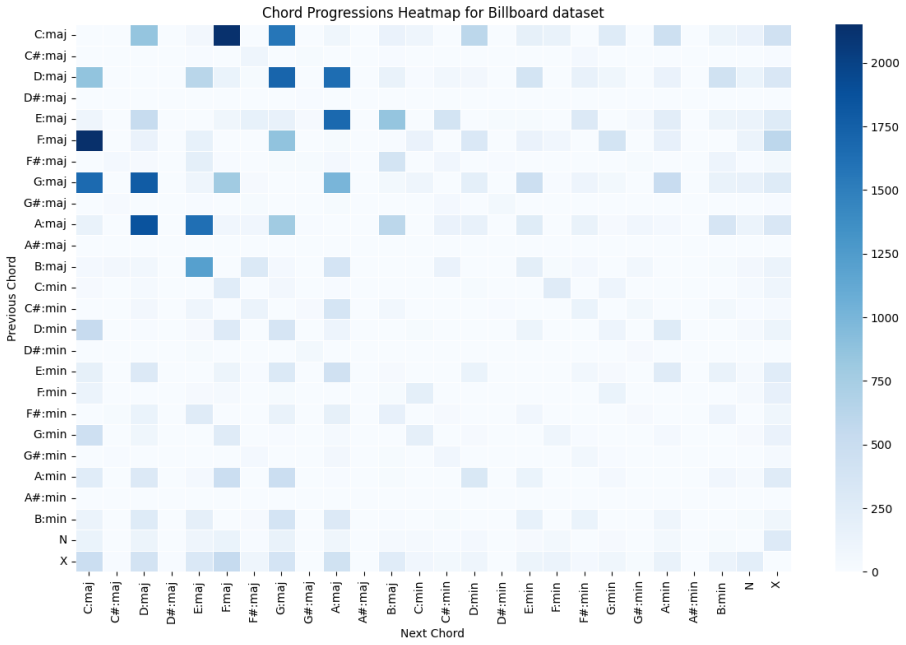


Figure 3.6: Number of changes from one chord to another in the Billboard dataset. Only changes to a different chord are counted.



3.3. Data preprocessing

For both model architectures, the data was preprocessed in the same way as the data in the papers introducing them.

In BTC, each 10-second audio signal (with consecutive signals overlapping by 5 seconds) was processed at a sampling rate of 22,050 Hz using CQT, covering 6 octaves starting from C1, with 24 bins per octave and a hop size of 2048. The CQT features were then converted to log amplitude. Global z-normalization was applied, using the mean and variance calculated from the training data. Additionally, pitch augmentation was performed on the audio files with corresponding adjustments to the labels to reflect the pitch changes. Pitch augmentation ranging from -5 to +6 semitones was applied to all the training data.

For HT, for each track from AAM and Winterreise the NNLS chroma features were computed using the Chordino VAMP plugin [3], installed into the Sonic Visualizer software [4]. Because this was a highly manual process, for AAM a subset of 192 out of the 3000 songs was taken. The AAM tracks, which were saved in 44100Hz, were preprocessed with default settings (windows size 16384, window increment 2048), resulting in time frames of around 0.046 seconds. However, because Winterreise tracks are saved in 22050Hz, they needed to be processed with window size

[3] http://www.isophonics.net/nnls-chroma

[4] https://sonicvisualiser.org/

of 8192 and window increment of 1024 to achieve the same result. If both cases the files were saved to include the timestamp in seconds before the feature, which contains 12 treble chroma and 12 bass chroma. Each input sequence for the Harmony Transformer consists of 100 segments (approximately 23 seconds), created using a sliding window with a frame size of 21 and a hop size of 5. All the training data are augmented by shifting the pitch in both the input data and the annotation, expanding the training set 12 times.

Furthermore, annotations for Winterreise came in .csv format, and annotations for AAM in .arff format, so they each needed to be converted to .lab using functions created for this purpose.

model	dataset	total songs	total instances (train and valid)
BTC	AAM	3000	1 072 908
	Winterreise	48	18 312
HT	AAM	192	13 766
	Winterreise	48	3 775
	Billboard	739	73 550

Table 3.4: Number of songs and training/validation sequences in preprocessed dataset for each model

3.4. Experimental setup

The main focus of the experiments was to check how useful the artificially generated dataset (AAM) is in training chord recognition systems and how complex it is compared to the datasets containing natural music. A total of 13 experiments (training cycles) were conducted, each with a different training set. Additionally, we present results of the pretrained BTC without any additional training as experiment number 0. Each experiment was conducted using 6-fold crossvalidation. The songs were not shuffled when creating the split into training and validation set, so that in Winterreise 2 recordings of the same song always ended up in the same set.

For the experiments on BTC, there were two approaches: training the entire model from scratch and finetuning the final fully-connected layer in the pretrained model, provided by the authors. During finetuning weights in all layers except this last one were frozen. Authors of HT do not share the weights of a pretrained model, therefore only the first approach is used for it.

For all of the datasets, the annotations with MajMin vocabulary were used, containing the 12 Major chords, 12 Minor chords and the non-chord symbol N. BTC has two versions: for a standard vocabulary (25 classes) and large vocabulary (170 classes), so for the experiments the first version was used.

Experiment no	Model	Training datasets	Evaluation datasets
0	pretrained BTC	-	AAM, Winterreise
1	BTC	AAM	
2		Winterreise	
3		AAM, Winterreise	
4	pretrained BTC	AAM	
5		Winterreise	
6		AAM, Winterreise	
7	HT	Billboard	Billboard, AAM, Winterreise
8		AAM	
9		Winterreise	
10		Billboard, AAM	
11		Billboard, Winterreise	
12		AAM, Winterreise	
13		Billboard, AAM, Winterreise	

Table 3.5: Training and evaluation datasets for conducted experiments

Winterreise consists of 48 audio tracks of 24 different songs, AAM has 3000 tracks and Billboard 890 tracks, 739 of which are used when training the HT. For the experiments, where there is only one training dataset (1, 2, 3, 7, 8 and 9), the entire available dataset is used. In other cases, to create a balanced training dataset, for the AAM and Billboard datasets a subset of 192 song were taken, and each of the 48 songs in the Winterreise dataset was repeated in the training set 4 times.

Table 3.6 presents the values of training hyperparameters used during the experiments. The HT has an additional chord class 'X', which indicates an unrecognized chord.

	BTC	HT
dropout	0.2	0.5
learning rate	0.0001	0.0001
batch size	32	60
n classes	25	26

Table 3.6: Training hyperparameters

The BTC experiments were conducted using max epochs of 100, and the HT experiments with 100000 training steps. However, in both cases the training was stopped early if the validation accuracy did not improve for 10 epochs. Model checkpoints were saved each time the accuracy

improved, as well as every 40 epochs, regardless of improvement.

3.5. Metrics

For all experiments, WCSR using Root and MajMin metrics and Weighted Accuracy using the CCM were used as evaluation metrics. They are all described in more detail in section 2.4. It is worth noting that CCM is the least strict metric when it comes to defining the meaning of chords being equal and MajMin is the strictest. Because of that, for every validation dataset the relationship CCM > Root > Majmin will be maintained.

3.6. Implementation

The code used for all experiments and result analysis is available on a Github repository[5]. As mentioned before, this work uses the original implementations of both architectures, and the implementation of the experiment is based on the code shared by the authors in respective Github repositories [6] [7]. However, there were many adjustments and additions needed to conduct the desired experiments. For both repositories, the preprocessing functions were adjusted to used the paths for AAM and Winterreise datasets and mechanism of creating a dataset with multiple data sources, subsets of those datasets and training data multipliers was added. The scripts used to analyze the datasets and results and create plots are also available on the repository.

3.6.1. BTC

In this repository, scripts that convert .arff and .csv annotations into .lab format were added. During this process, a BASS NOTE EXCEPTION was found in place of a valid chord annotation in some AAM tracks, all of which were replaced by an 'N' chord label. Additionally, because AAM tracks are in .flac format, an implementation of function that finds the lengths of a file in this format [8] was added. There were many functions created for the large vocabulary version of BTC which were not used in this work, so they were deleted. The implementation of CCM needed to be added to the metric implementation and modified to be able to process 'N' chords.

[5] https://github.com/m-majchrzak/Automatic_Audio_Chord_Recognition
[6] https://github.com/jayg996/BTC-ISMIR19
[7] https://github.com/Tsung-Ping/Harmony-Transformer
[8] https://gist.github.com/lukasklein/8c474782ed66c7115e10904fecbed86a

3.6.2. HT

The HT repository uses Tensorflow 1, so the code needed to be adjusted to be compatible with Tensorflow 2. This repository also had a much less extensive training framework, so the entire mechanism of calculating validation loss and accuracy, early stopper, logging, checkpoint saving and metric implementation was added.

3.6.3. Computational resources

All experiments were conducted using the computational resources of the $Eden^N$ HPC Cluster[9] at Faculty of Mathematics and Information Science, Warsaw University of Technology. Because of that, the batch scripts for executing certain Python scripts on the cluster are also present in the repository, specifying queue, time and resources for a certain task.

[9] https://hpc.mini.pw.edu.pl/

4. Results

In this chapter, results from all experiments are presented. Section 4.1 presents results from experiments number 0 - 6, conducted with the BTC model architecture, and section 4.2 experiments 7 - 13, conducted with the HT model architecture.

4.1. Experiments on BTC

Figure 4.1: Evaluation metrics on AAM as a validation dataset for all experiments on BTC model

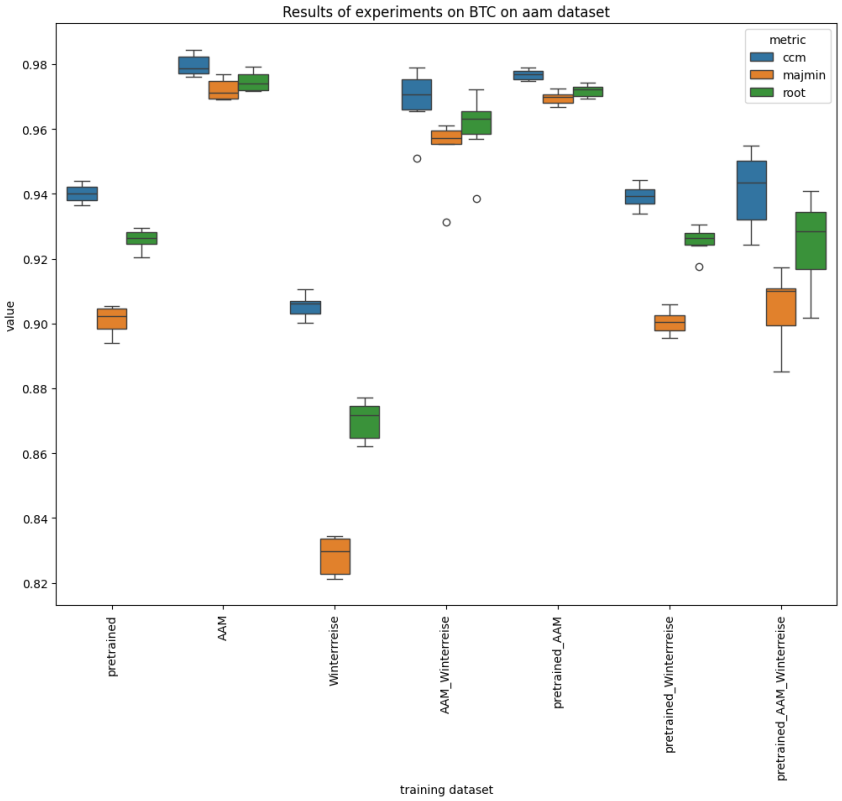


Figure 4.1 shows the values of metrics on the AAM dataset. The results have quite small standard deviations, suggesting that the songs in this dataset have a fairly even complexity among all 6 folds. The model trained only on Winterreise has the lowest values of all metrics.

The performance of the pretrained models that were not trained on AAM (pretrained and pretrained_Winterreise) is much better, suggesting that a lot of the complexity of the AAM dataset was captured by the corpus of pop songs that BTC was trained on. The pretrained model fine-tuned on both AAM and Winterreise (pretrained_AAM_Winterreise) is performing worse than the one trained on the same corpus from scratch.

Figure 4.2: Evaluation metrics on Winterreise as a validation dataset for all experiments on BTC model

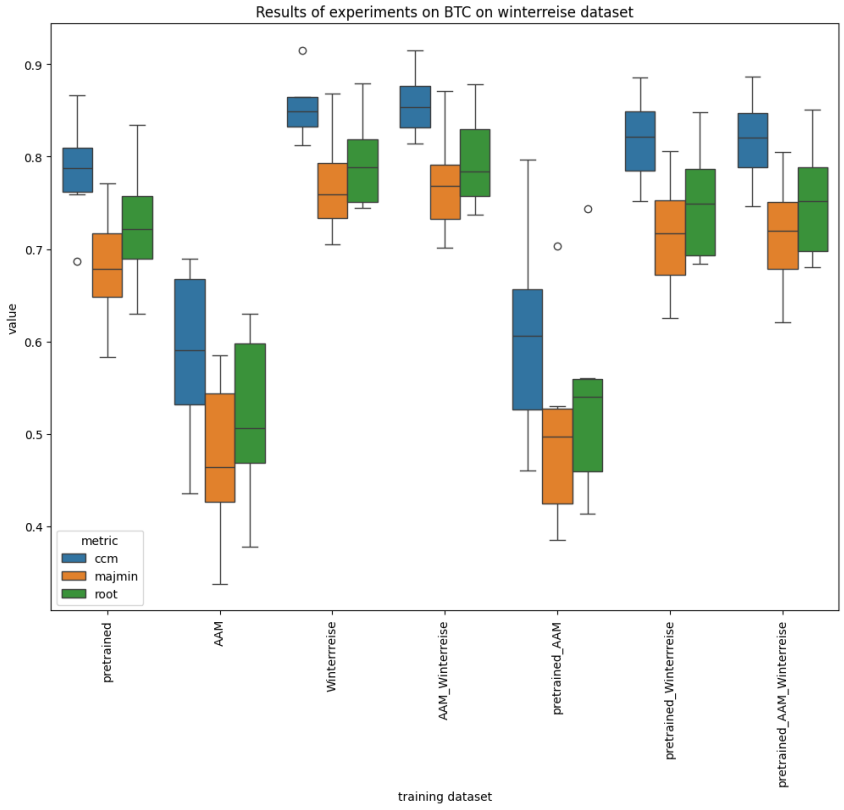


Figure 4.2 shows the values of metrics on Winterreise dataset. The values are, in general, lower than in figure 4.1, suggesting that this dataset is more difficult to predict than the artificially created one and they have higher standard deviation, which means songs in some folds were much more difficult to predict than others. Additional investigation revealed that songs in fold number 4 (songs with numbers Schubert_D911-17 to Schubert_D911-20) seem to be the most difficult. Unsurprisingly, the models that were not trained on Winterreise (AAM and pretrained_AAM) have the lowest values of all metrics. For this evaluation dataset it does not make a huge difference whether the model was trained from scratch or fine-tuned, which would suggest that weights learned on the corpus of pop songs that BTC was pretrained do not help the model correctly predict chords in Winterreise as much as they did for AAM.

The mean and standard deviation of metric values presented of the plot above are summarized in table 4.1. It is again clearly visible that results on Winterreise have lower means and higher

Exp id	AAM			Winterreise		
	Root	MajMin	CCM	Root	MajMin	CCM
0	92.59 ± 0.33	90.11 ± 0.46	94.01 ± 0.29	72.59 ± 7.05	67.99 ± 6.61	78.29 ± 6.06
1	**97.47** ± 0.32	**97.22** ± 0.34	**97.97** ± 0.35	51.74 ± 9.69	47.25 ± 9.26	58.54 ± 9.88
2	87.01 ± 0.63	82.84 ± 0.62	90.54 ± 0.37	79.46 ± 5.26	77.02 ± 5.92	85.35 ± 3.63
3	96.01 ± 1.17	95.36 ± 1.12	96.88 ± 1.00	**79.64** ± 5.46	**77.15** ± 5.99	**85.74** ± 3.70
4	97.18 ± 0.20	96.96 ± 0.21	97.68 ± 0.17	53.92 ± 11.73	50.33 ± 11.37	60.71 ± 11.99
5	92.55 ± 0.45	90.04 ± 0.38	93.92 ± 0.38	75.06 ± 6.58	71.47 ± 6.64	81.85 ± 4.99
6	92.48 ± 1.47	90.50 ± 1.20	94.12 ± 1.22	75.26 ± 6.58	71.50 ± 6.58	81.77 ± 5.08

Table 4.1: Mean and standard deviation from results of 6 fold validation from experiments on BTC model. The largest mean value in each column is bolded.

standard deviations. The highest values for evaluation on AAM were achieved by a model trained from scratch on AAM (experiment number 1) and the highest values for evaluation on Winterreise were achieved by a model trained from scratch on both AAM and Winterreise (experiment number 3), however the difference between this one and the one trained just on Winterreise (experiment number 2) is not significant.

4.2. Experiments on HT

Figure 4.3: Evaluation metrics on AAM as a validation dataset for all experiments on HT model

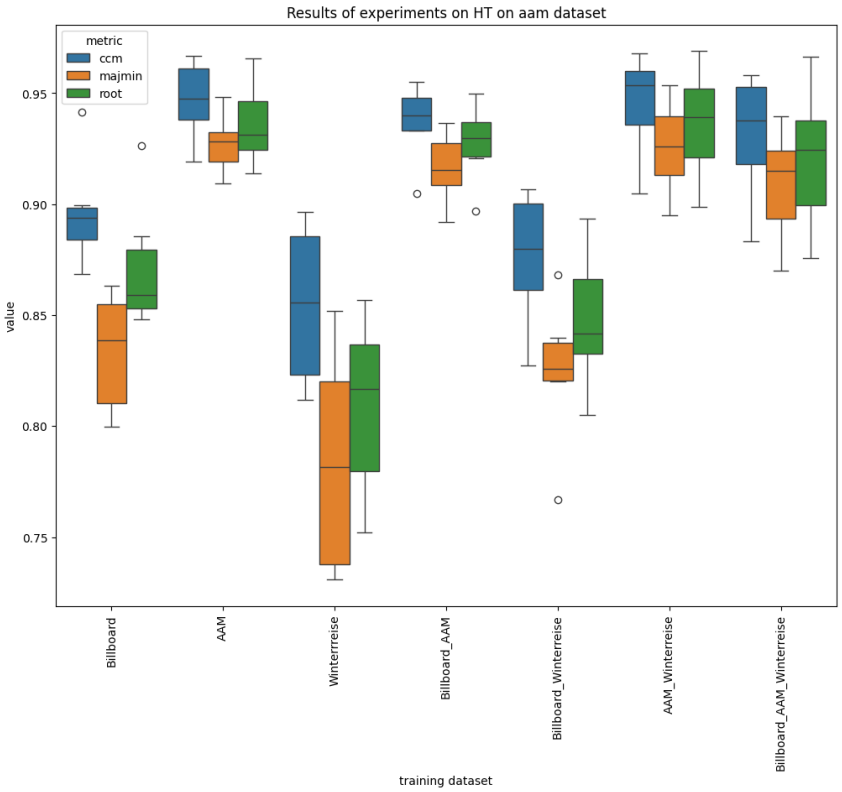
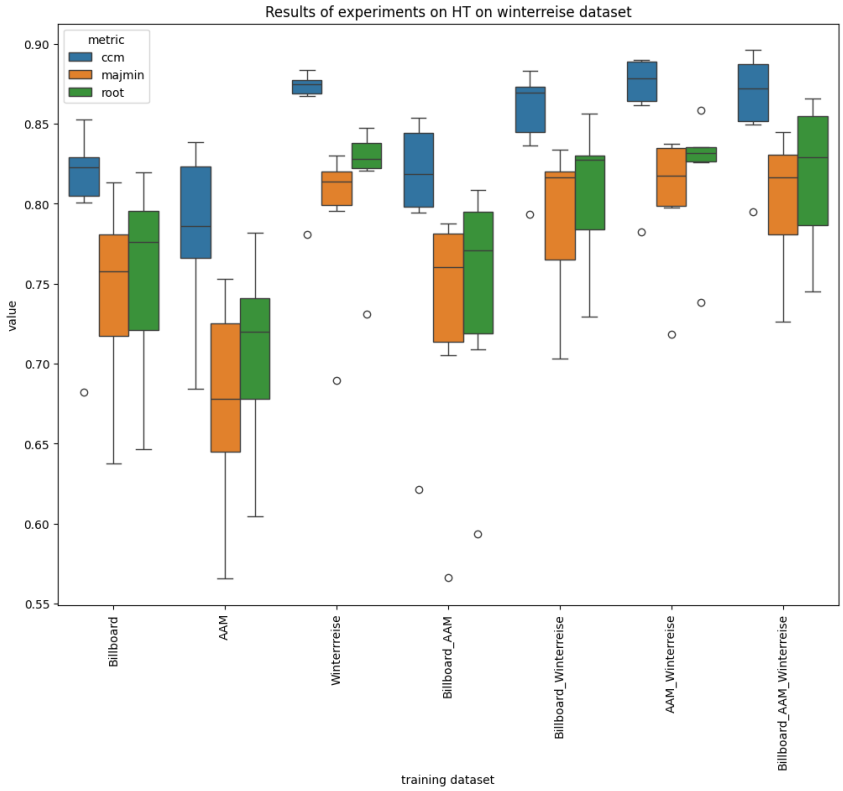


Figure 4.3 shows the values of metrics on the AAM dataset. The performance of the model trained from scratch on Billboard is better than the one trained on Winterreise, which would suggest that Billboard and AAM are more similar than Winterreise and AAM.

Figure 4.4: Evaluation metrics on Winterreise as a validation dataset for all experiments on HT model

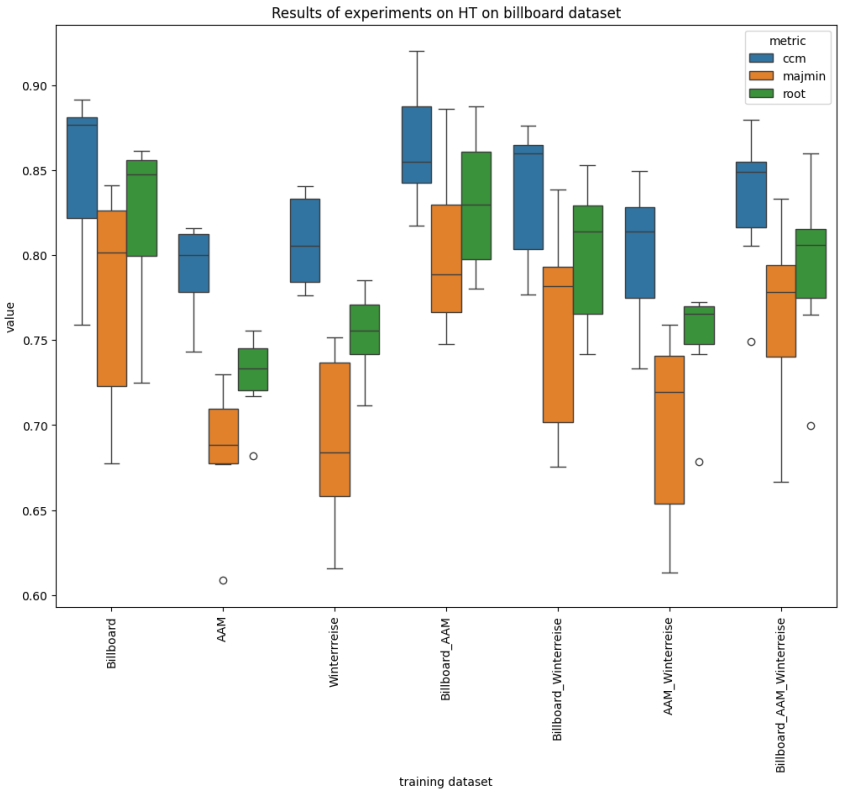


Values of metrics on Winterreise dataset for all experiments on HT model are presented in figure 4.4. The outliers visible are again the results on fold number 4. The model trained on Billboard is performing slightly better than the one trained on AAM. Adding AAM to the training dataset does not seem to improve performance (comparing Billbaord to Billboard_AAM, Winterreise to AAM_Winterrise and Billboard_Winterreise to Billboard_AAM_Winterreise).

Figure 4.5 shows the values of metrics on the Billboard dataset. The results of a model trained on AAM are comparable to the model trained on Winterreise and the one trained on both, so it could mean that AAM could be a decent training set for a model which purpose is to predict chord sequences in pop music, if no other data is available. In this case, adding a big artificial corpus to a small set of classical music (comparing Winterreise to AAM_Winterreise) does seem to improve performance. That could indicate that in the sense of chord sequence similarity, AAM and Billboard are more similar than AAM and Winterreise.

The mean and standard deviation of metric values presented of the plot above are summarized in table 4.2. For predicting chord sequences in AAM, the model trained on AAM (experiment

Figure 4.5: Evaluation metrics on Billboard as a validation dataset for all experiments on HT model



number 8) or AAM_Winterreise (experiment number 12) were the best, depending on the evaluation metric. In terms of predicting Winterreise, the AAM_Winterreise (experiment number 12) model was the best, and for Billboard - Billboard_AMM model (experiment number 10).

Exp id	AAM			Winterreise			Billboard		
	Root	MajMin	CCM	Root	MajMin	CCM	Root	MajMin	CCM
7	87.16 ± 3.00	83.35 ± 2.74	89.64 ± 2.48	75.40 ± 6.53	74.31 ± 6.28	80.18 ± 6.09	82.04 ± 5.47	77.55 ± 7.01	84.83 ± 5.39
8	93.59 ± 1.90	**92.74** ± 1.35	**94.68** ± 1.83	70.64 ± 6.26	67.50 ± 6.85	78.17 ± 5.65	72.83 ± 2.64	68.44 ± 4.22	79.10 ± 2.84
9	80.88 ± 4.10	78.38 ± 5.20	85.45 ± 3.69	81.59 ± 4.27	79.40 ± 5.24	85.98 ± 3.90	75.36 ± 2.63	68.99 ± 5.40	80.78 ± 2.84
10	92.73 ± 1.82	91.61 ± 1.61	93.69 ± 1.80	74.13 ± 8.13	72.69 ± 8.50	79.28 ± 8.69	**83.09** ± 4.22	**80.23** ± 5.23	**86.39** ± 3.83
11	84.77 ± 3.15	82.44 ± 3.32	87.60 ± 3.06	80.69 ± 4.74	78.98 ± 5.22	85.43 ± 3.38	80.10 ± 4.52	75.82 ± 6.67	83.73 ± 4.39
12	**93.62** ± 2.56	92.55 ± 2.14	94.54 ± 2.35	**82.04** ± 4.17	**80.39** ± 4.54	**86.34** ± 4.13	74.90 ± 3.63	69.84 ± 6.00	80.09 ± 4.38
13	92.07 ± 3.27	90.88 ± 2.58	93.09 ± 2.87	81.74 ± 4.85	80.15 ± 4.47	86.21 ± 3.76	79.24 ± 5.46	76.39 ± 5.83	83.15 ± 4.70

Table 4.2: Mean and standard deviation from results of 6 fold validation from experiments on HT model. The largest mean value in each column is bolded.

5. Conclusions

Based on the results described above, one can conclude that even though the AAM dataset is artificially generated, it can be useful in certain settings for training and evaluating neural network models for Audio Chord Recognition.

Conducted experiments suggest that the chord progressions in AAM dataset are the easiest to predict for both model architectures, followed by Billboard being more complicated, and then Winterreise being the hardest. This might be due to the fact that both Billboard and Winterreise have annotations with a larger, more complicated vocabulary, and those experiments use the smaller, simplified one. For AAM, only a MajMin vocabulary is present, so the segment labeled as a certain chord is exactly that, without any additional notes.

For the Bidirectional Transformer for Chord Recognition, training from scratch generally worked better than fine-tuning a pretrained model in the scenario where the training and evaluation datasets were the same. However, starting from pretrained weights improved the evaluation of the model trained on Winterreise on the AAM dataset.

For the Harmony Transformer, enriching the training dataset with AAM improved performance for both Winterreise and Billboard by about 1 percentage point.

BTC and HT as neural network models for Audio Chord Recognition based on the Transformer architectures can achieve comparable results, all depending on the training and validation datasets.

In future works, one could consider automating the process of creating NNLS chroma features using the Chordino plugin and process the entire AAM dataset instead of a subset, and repeat the experiments using HT. Another possible improvement would be recreating the dataset used for training BTC by downloading songs from streaming services and including them in similar experiments.

Bibliography

[1] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, volume 2, page 335340, 2013.

[2] Judith C. Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 01 1991.

[3] John Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. pages 633–638, 01 2011.

[4] Tristan Carsault, Jérôme Nika, and Philippe Esling. Using musical relationships between chord labels in automatic chord extraction tasks. 09 2018.

[5] Tsung-Ping Chen and Li Su. Harmony transformer: Incorporating chord segmentation into harmony recognition. In *International Society for Music Information Retrieval Conference*, 2019.

[6] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, 01 2014.

[7] Taemin Cho and Juan P. Bello. On the relative importance of individual components of chord recognition systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):477–492, 2014.

[8] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation using bidirectional long short-term memory recurrent neural network with even chance training. *Journal of New Music Research*, 47:1–15, 10 2017.

[9] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation using deep neural nets: Design framework, system variations and limitations. 09 2017.

[10] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation with even chance training scheme. 2017.

[11] Johanna Devaney. Beyond chord vocabularies: Exploiting pitch-relationships in a chord estimation metric. *CoRR*, abs/2201.05244, 2022.

[12] Takuya Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *International Conference on Mathematics and Computing*, 1999.

[13] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.

[14] Christopher Harte. *Towards Automatic Extraction of Harmony Information from Musc Signals*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London, august 2010.

[15] Christopher Harte, Mark Sandler, Samer Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. pages 66–71, 01 2005.

[16] Takeshi Hori, Kazuyuki Nakamura, and Shigeki Sagayama. Music chord recognition from audio data using bidirectional encoder-decoder lstms. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1312–1315, 2017.

[17] Eric J. Humphrey and Juan P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 357–362, 2012.

[18] Eric J. Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *International Society for Music Information Retrieval Conference*, 2015.

[19] Junyan Jiang, Ke Chen, Wei Li, and Guangyu Xia. Mirex 2018 submission: A structural chord representation for automatic large-vocabulary chord transcription. MIREX 2018, 2018.

[20] H. V. Koops, W. B. de Haas, J. Bransen, and A. Volk. Chord label personalization through deep learning of integrated harmonic interval-based representations, 2017.

[21] Hendrik Vincent Koops, W. Bas de Haas, John Ashley Burgoyne, Jeroen Bransen, Anna Kent-Muller, and Anja Volk. Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, 48(3):232–252, 2019.

[22] Hendrik Vincent Koops, Gianluca Micchi, Ilaria Manco, and Elio Quinton. Serenade: A model for human-in-the-loop automatic chord estimation. *2023 4th International Symposium on the Internet of Sounds*, pages 1–7, 2023.

[23] Filip Korzeniowski, David Sears, and Gerhard Widmer. A large-scale study of language models for chord prediction. 04 2018.

[24] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. 08 2016.

[25] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. *CoRR*, abs/1612.05082, 2016.

[26] Filip Korzeniowski and Gerhard Widmer. On the futility of learning complex frame-level language models for chord recognition. 02 2017.

[27] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. *ArXiv*, abs/1808.05335, 2018.

[28] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models, 2018.

[29] Song-Rong Lee, I Chien, Tzu-Chun Yeh, and Jyh-Shing Roger Jang. Mirex 2019 submission: Chord estimation. MIREX 2019, 2018.

[30] Matthias Mauch. *Automatic Chord Transcription from Audio Using Computational Models of Musical Context*. PhD thesis, School of Electronic Engineering and Computer Science Queen Mary, University of London, march 2010.

[31] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. pages 135–140, 01 2010.

[32] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *International Society for Music Information Retrieval Conference*, 2017.

[33] Matt McVicar, Raúl Santos-Rodríguez, Yizhao Ni, and Tijl De Bie. Automatic chord estimation from audio: A review of the state of the art. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):556–575, 2014.

[34] M. Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, 2016.

[35] S.H. Nawab, S.A. Ayyash, and R. Wotiz. Identification of musical chords using constant-q spectra. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 5, pages 3373–3376 vol.5, 2001.

[36] Julien Osmalskyj, Jean Jacques Embrechts, Marc Droogenbroeck, and Sébastien Piérard. Neural networks for musical chords recognition. 01 2012.

[37] Fabian Ostermann, Igor Vatolkin, and Martin Ebeling. Aam: a dataset of artificial audio multitracks for diverse music information retrieval tasks. *EURASIP Journal on Audio, Speech, and Music Processing*, 2023, 03 2023.

[38] Jonggwon Park, Kyoyun Choi, Sungwook Jeon, Dokyun Kim, and Jonghun Park. A bi-directional transformer for musical chord recognition. *CoRR*, abs/1907.02698, 2019.

[39] Johan Pauwels, Ken O'Hanlon, Emilia Gómez, and Mark B. Sandler. 20 years of automatic chord recognition from audio. In *International Society for Music Information Retrieval Conference*, 2019.

[40] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 749–753, 2013.

[41] Colin Raffel, Brian Mcfee, Eric Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel Ellis. mir$_e$*val* : *Atransparentimplementationofcommonmirmetrics.*102014.

[42] Ricardo Scholz, Emmanuel Vincent, and Frederic Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 53–56, 2009.

[43] Hendrik Schreiber, Christof WeiSS, and Meinard Müller. Local key estimation in classical music recordings: A cross-version study on schuberts winterreise. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 501–505, 2020.

[44] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *International Society for Music Information Retrieval Conference*, 2015.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[46] Christof Weiss and Geoffroy Peeters. Training deep pitch-class representations with a multi-label ctc loss. In *22nd International Society for Music Information Retrieval Conference*, page 754761, 2021.

[47] Christof Weiß, Frank Zalkow, Vlora Arifi-Müller, Meinard Müller, Hendrik Vincent Koops, Anja Volk, and Harald G. Grohganz. Schubert winterreise dataset: A multimodal scenario for music analysis. *J. Comput. Cult. Herit.*, 14(2), may 2021.

[48] Christof Weiss, Johannes Zeitler, Tim Zunner, Florian Schuberth, and Meinard Müller. Learning pitch-class representations from score-audio pairs of classical music. In *22nd International Society for Music Information Retrieval Conference*, page 74653, 2021.

[49] Yiming Wu, Tristan Carsault, Eita Nakamura, and Kazuyoshi Yoshii. Semi-supervised neural chord estimation based on a variational autoencoder with discrete labels and continuous textures of chords. *CoRR*, abs/2005.07091, 2020.

[50] Yiming Wu, Tristan Carsault, and Kazuyoshi Yoshii. Automatic chord estimation based on a frame-wise convolutional recurrent neural network with non-aligned annotations. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[51] Yiming Wu and Wei Li. Automatic audio chord recognition with midi-trained deep feature and blstm-crf sequence decoding model. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, PP:1–1, 11 2018.

[52] Yiming Wu and Wei Li. Music chord recognition based on midi-trained deep feature and blstm-crf hybird decoding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 376–380, 2018.

[53] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. 01 2015.

List of symbols and abbreviations

ACE	Automatic Chord Estimation
ACR	Automatic Chord Recognition
BLSTM	Bidirectional Long Short-Term Memory
CMM	Chord Content Metric
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
CRF	Conditional Random Field
CQT	Constant Q Transform
DBN	Dynamic Bayesian Networks
DFT	Discrete Fourier Transform
GMM	Gaussian Mixture Modelling
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MIREX	Music Information Retrieval Evaluation eXchange
NNLS	Non-Negative Least Squares
OR	Overlap Ratio
PCP	Pitch Class Profile
RNN	Recurrent Neural Network
STFT	Short-Time Fourier Transform
WAOR	Weighted Average Overlap Ratio
WCSR	Weighted Chord Symbol Recall
VAE	Variational Autoencoder

List of Figures

List of Tables