Lead Sheet Transcription

Pierre Lardet

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2025

Abstract

This skeleton demonstrates how to use the `infthesis` style for undergraduate dissertations in the School of Informatics. It also emphasises the page limit, and that you must not deviate from the required style. The file `skeleton.tex` generates this document and should be used as a starting point for your thesis. Replace this abstract text with a concise summary of your report.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Pierre Lardet*)

Acknowledgements

Table of Contents

Chapter 1

Introduction

1.1 Motivation

TODO: Rewrite introduction to focus on chord recognition.

- Useful for musicians - Useful for musicologists - Connection to lead sheets

1.2 Aims

The aims of this project are:

- Compare state-of-the-art models for automatic chord recognition.
- Investigate methods of improving on this baseline model for music
- To investigate the use of synthetic data generation for improving the performance of the model.

1.3 Outline

The report is structured as follows:

- **Chapter 2** provides background information on chord transcription and related work.
- **Chapter 3** describes the datasets, evaluation metrics and training procedure used in this project.
- **Chapter 4** compares various models from the literature and investigates improvements.
- **Chapter 5** extends this work with synthetic data generation and compares results on a new dataset.
- **Chapter 6** concludes the report and provides suggestions for future work.

Chapter 2

Background & Related Work

2.1 Background

TODO: Rewrite background to focus on background chord recognition.

- Chords, harmony, ambiguity and importance - Chord recognition as a subfield of music transcription and older related papers. - Connection to lead sheets, definition.

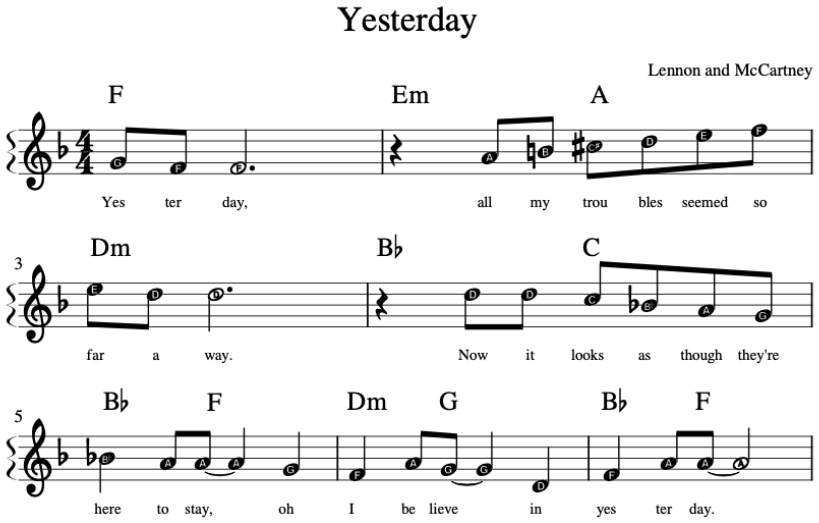

Figure 2.1: An example of a lead sheet for 'Yesterday' by the Beatles.

2.1.1 Music Features

Recorded music can be represented in a variety of ways as input to a machine learning model. The simplest way is to leave the data as a waveform: a time-series vector of amplitudes. Data in the raw audio domain has seen successful use in generative models such as Jukebox (Dhariwal et al., 2020) and RAVE (Caillon and Esling, 2021).

Spectrogram: A very commonly used representation of general audio data, and specifically musical data, is the spectrogram. A spectrogram is a conversion of the time-series

data into the time-frequency domain, calculated by a SFTF (short-time Fourier Transform) Spectrograms are commonly used in many audio processing tasks, such as speech recognition, music recognition (Wang, 2003) and music transcription, specifically polyphonic transcription (Toyama et al., 2023). However, as noted by Pauwels et al. (2019), only logarithmic spectrograms have been used in ACR tasks, and linear spectrograms have been used in melody transcription tasks.

CQT: A common version of the spectrogram used in music transcription is the Constant-Q Transform (CQT), originally proposed by Brown (1991). The CQT is a version of a spectrogram with frequency bins that are logarithmically spaced and bin widths that are proportional to the frequency. This is motivated by the logarithmic nature of how humans perceive pitch: a sine wave that is perceived as one octave higher than another has double the frequency. The CQT is used in many music transcription tasks, such as automatic chord recognition (Humphrey and Bello, 2012), and a popular alternative, Melspectrograms, have found use in melody transcription (Toyama et al., 2023). An example CQT from the dataset used in this work is shown in Figure 2.2.

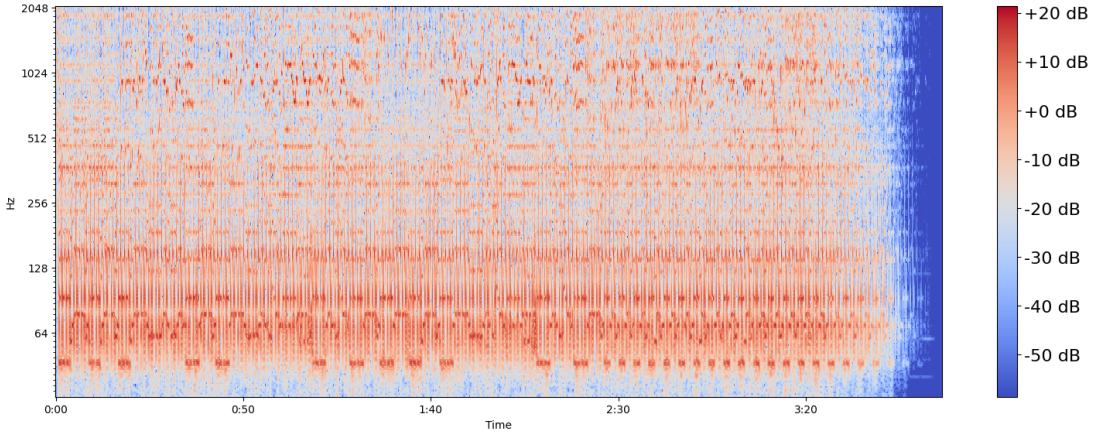

Figure 2.2: A sample CQT of 'Girls Just Wanna Have Fun' by Cyndi Lauper from the dataset used in this work. We can clearly see the log-spaced frequency bins. We also observe clear structure and repetition in the song, espeically in the lower frequencies, which can be attributed to a regular drum groove and bass instruments. This is typical of pop songs in this dataset.

Chroma Vectors: Chroma vectors are a 12-dimensional time-series representation, where each dimension corresponds to a pitch class. Each element represents the presence of each pitch class in the Western chromatic scale in a given time frame. Such features have been generated by deep learning methods (Miller et al., 2022) or by hand-crafted methods (Mauch and Dixon, 2010) and have seen use in recent ACR models (Chen and Su, 2019).

Generative Features: More recently, features extracted generative models have been used as input. The proposed benefit is that the vast quantities of data used to train these models allows for rich representations of the music. Chris Donahue and Liang (2022) use features from JukeBox (Dhariwal et al., 2020) to train a transformer (Vaswani et al., 2023) for both melody transcription and chord recognition and combine these into a

lead sheet generation model. They found that these features outperformed hand-crafted features in melody transcription tasks but did not perform experiments in ACR.

2.2 Related Work

2.2.1 Automatic Chord Recognition

Pauwels et al. (2019) provides an overview of ACR since its inception in 1999 with the work of Fujishima (1999) up to 2019, and provide suggestions for future avenues of research.

Datasets: Source of data that have seen common use in ACR relevant to this work include:

- *Mcgill Billboard*: 890 chord annotations of songs randomly selected from the Billboard 'Hot 100' Chart bnetween 1958 anmd 1991. (Burgoyne et al., 2011)
- *Isophonics*: 300 annotations of songs from albums by The Beatles, Carole King and Zweieck. (Cannam et al., 2009)
- *RWC-Pop*: 100 pop songs with annotations available[1] for chords. (Goto et al., 2002)
- *USPop*: 195 annotations of a larger dataset with artists chosen for popularity. (Berenzweig et al., 2004)
- *JAAH*: 113 annotations of a collection of jazz recordings. Durán and de la Cuadra (2020)
- *HookTheory*: 50 hours of labelled audio in the form of short musical segments, crowdsourced from the HookTheory website[2]. (Chris Donahue and Liang, 2022)

Other small datasets also exist. Many of these have been compiled together into the *Chord Corpus* by de Berardinis et al. (2023), with standardised annotation formats.

A big problem frequently encountered in ACR is the lack of labelled data. This work uses 1200 labelled songs with audio. This is due to the difficulty and time associated with labelling data aligned in time and the legal sensitivity of the data involved.Data has been scaled up using augmentation and semi-supervised learning (Hung et al., 2023) with some success. Research has been done into the use of synthetic data (Kroher et al., 2023; Sato and Akama, 2024) and supervised learning (Li et al., 2024) for MIR tasks, but not for ACR.

Another problem is that the existing data is often imbalanced, with a large number of common chords like major and minor chords and fewer chords like diminished and augmented chords, or chords with upper extensions and inversions. This can lead to models that are biased towards predicting major and minor chords. Attempts to

[1] `https://github.com/tmc323/Chord-Annotations`
[2] https://www.hooktheory.com/

address this have been made by re-weighting classes (Jiang et al., 2019) or adjusting the sampling of training examples to balance chord classes (Miller et al., 2022).

Models: A variety of machine learning methods have been applied to such datasets. Older methods such as Lee and Slaney (2006) use HMMs, but more recent methods use deep learning. Often, CNNs and RNNs are used in combination (Wu et al., 2019; Jiang et al., 2019; McFee and Bello, 2017) with CNNs performing feature extraction from a spectrogram feature, and an RNN (normally Bi-LSTM or Bi-GRU) predicts chord sequences. More recently, transformers have been applied to the entire process Chris Donahue and Liang (2022) and Chen and Su (2019, 2021).

Evaluation: Evaluation is typically done using accuracy and recall of correct chord predictions, or more music-aware measures such as the correct root note, 3rd or the MIREX metric which measures that chords have at least 3 notes in common. These are all implemented by **?** in the `mir_eval` library [3]. Qualitative evaluation is also often carried out.

2.2.2 Lead Sheet Transcription

Older work has attempted to automatically produce lead sheets (Ryynänen and Klapuri, 2008; Weil et al., 2009). They use hand-crafted feature extractions and probabilistic models for melody and chord classification. AMT has developed dramatically since this work which leaves much room for improvement.

(Chris Donahue and Liang, 2022) is the only recent work we found that produces a full lead sheet generation model. They propose the use of features extracted from middle layers if Jukebox (Dhariwal et al., 2020) which were found to lead to the best performance in downstream tasks (Castellon et al., 2021). These features are used to train a transformer, with a primary focus on melody transcription. The model and code is available[4]. They use the same methodology to train an ACR model, and combine these with beat detection and engraving software to produce a lead sheet.

The authors found the use of these generative pre-trained features to be beneficial for melody transcription. However, they use community-submitted annotations from an online forum, presumably for lack of a better alternative, whose quality and variety may be lacking. The nature of this data means that the model is only trained on 24 second audio clips, which limits performance on longer segments of audio. The authors claim that the overall model performs well, especially in the verses and choruses of pop music where vocal lines are loud and clear, and that performance across genres is surprisingly good given the pop-centric dataset. However, further analysis is ommitted and if the dataset contains music with few songs with more complex harmonic structures such as jazz music, then the model will produce overly simplistic representations of the chords. Crucially, the work does not explore ACR models beyond simply copying their method and dataset used in melody transcription and there is no quantitative evaluation of performance in ACR. Furthermore, the system can struggle with melody lines shifting between instruments, quiet vocal lines

[3] `https://mir-evaluation.github.io/mir_eval/`

[4] `https://github.com/chrisdonahue/sheetsage/tree/main`

Chapter 3

Experimental Setup

This chapter outlines the datasets used in this work, the preprocessing applied to the audio and chord annotations, the evaluation metrics used to compare the models and details of the training process used throughout.

3.1 Datasets

Two ACT datasets are used in this work. The first dataset is simply referred to as the *Pop* dataset, as much of the music in the dataset comes from the Billboard Hot 100 charts, or other popular bands. The second dataset is the *JAAH* (Jazz Annotations and Analysis of Harmony) dataset mentioned in Section 2.2.1. While the chords annotations are publicly available on Github[1], the audio was kindly given to me by Andrea Poltronieri, a PhD student at the University of Bologna and the author of 'ChoCo' (the Chord Corpus) (de Berardinis et al., 2023). I spent the first weeks of the project contacting authors of several papers which have used these or other datasets and he was the only one helpful enough to provide me with the audio.

The rest of this chapter explains processing applied to the audio and chord annotations common to both datasets, before discussing details of the *Pop* and *JAAH* datasets relevant to each.

3.1.1 Preprocessing

3.1.1.1 Audio to CQT

The audio was first converted to a Constant-Q Transform (CQT) representation explained in Section 2.1.1. The CQT is computed using the `librosa` library (McFee et al., 2015), using the `.cqt` function. A sampling rate of 44100Hz was used, with a hop size of 4096, and 36 bins per octave, 6 octaves and a fundamental frequency corresponding to the note `C1`. This returns a complex-valued matrix containing phase, frequency and amplitude information. Phase information is discarded by taking the absolute value, before being converted from amplitude to dB, equivalent to taking the logarithm. These

[1] https://github.com/smashub/choco

default parameters were chosen to be consistent with previous works (McFee and Bello, 2017). However, the hop size was later varied in order to investigate the effect of different frame lengths on performance. An alternative choice of frame length could be to use some fraction of a bar. For example, Chris Donahue and Liang (2022) use 1/16th of a bar. However, most work in the literature opts for a constant hop size. We follow this convention, as a hop size which varies over songs introduces a new source of error in identifying the beats of a song, which the authors note as sometimes being problematic to their method.

This leads to a CQT matrix of a song having dimensions $216 \times F$ where 216 is the number of frequency bins and F is the number of frames in the song. For context, the number of frames can be calculated as $F = \lceil \frac{44100}{4096} L \rceil$ where L is the length of the song in seconds, 44100 is the sampling rate in Hertz (Hz) and 4096 is hop size in samples. A 3 minute song has just under 2000 frames. To save on computational time, the CQT was pre-computed into a cached dataset rather than re-computing each CQT on every run. This was done for each hop size used.

3.1.1.2 Chord Annotations

The chord annotations are represented as a sorted list of dictionaries, each containing the chord, start time and duration. The chord itself is represented as a string in Harte notation (Harte et al., 2005). For example, C major 7 is `C:maj7` and A half diminished 7 in its second inversion is `A:hdim7/5`. However, the majority chords in the dataset are not this complicated. The notation also includes `N` representing no chord and `X` representing an out of gamut chord.

This annotation is far too flexible to be used as targets for an ML model. This would lead to thousands of chords. Instead, we define two chord vocabularies. The first is a simple chord vocabulary, which contains only major, minor for each root and a no chord symbol `N`. Then this vocabulary has 26 labels. Chords outside the vocabulary are mapped to `X`. For example, `C:maj7` would be mapped to `C:maj`, while `A:hdim7/5` would be mapped to `X`. The second is a more complex vocabulary, which contains 14 qualities for each root: major, minor, diminished, augmented, minor 6, major 6, minor 7, minor-major 7, major 7, dominant 7, diminished 7, half diminished 7, suspended 2, suspended 4. Additionally, there is a no chord `N` and an out-of-gamut symbol `X`, totalling $12 \cdot 14 + 2 = 170$ chord labels. This vocabulary or similar vocabularies have been used by much of the literature (McFee and Bello, 2017; Humphrey and Bello, 2015; Jiang et al., 2019).

Chords in Harte notation are mapped to this vocabualry by first converting them to a tuple of integers in the range 0-11 representing pitch classes using the Harte library. These are transposed that 0 is the root. These pitch classes are then matched to a of template identifying the quality of the chord, as decribed in (McFee and Bello, 2017). However, for some chords this was not sufficient. For example, a `C:maj6(9)` chord would not fit perfectly with any of these templates due to the added 9th. Therefore, the chord is also passed through Music21's (Cuthbert and Ariza, 2010) chord quality function which matches chords such as the one above to major. This function would not work alone as its list of qualities is not as rich as the one defined above. If the chord is

still not matched, it is mapped to `X`. This additional step is not done by McFee and Bello (2017) but gives more meaningful labels to roughly one third of the chords previously mapped to `X`.

3.1.2 Pop

The *Pop* dataset consists of songs from the *Mcgill Billboard*, *Isophonics*, *RWC-Pop* and *USPop* datasets mentioned in Section 2.2.1. This collection was originally proposed in work by Humphrey and Bello (2015) in order to bring together most of the known datasets for chord recognition. The dataset consists of 1,217 songs, filtered for duplicates and selected for those with annotations available. The dataset was provided with obfuscated filenames and audio as `.mp3` files and annotations as `.jams` files (Humphrey et al., 2014).

3.1.2.1 Data Integrity

Several possible sources of error in the dataset were investigated.

Duplicates: Files were renamed using provided metadata identifying them by artist and song title. This was done to identify duplicates in the dataset. The dataset was first filtered for duplicates, of which there was one - Blondie's 'One Way or Another', which had two different versions. The duplicate was removed from the dataset. Further duplicates may exist under different names but the songs were listened to throughout the project and no othe rduplicates were found. Automatic analysis of the audio could help, although cannot be guaranteed to find different versions of the same song, as above.

Chord-Audio Alignment: 10 songs were manually investigated for alignment issues. This was done by listening to the audio and comparing it to the annotations directly at various points in the song, with a focus on the beginning. It became apparent that precise timings of chord changes are ambiguous. The annotations aired on the side of being slightly early but were all certainly good annotations, with detailed chord labelings including inversions and upper extensions. This observation was borne in mind later when analysing performance of the models at transition frames in Section [XX].

Automatic analysis of the alignment of the audio and chord annotations was also done using cross-correlation of the derivative of the CQT features of the audio over time and the chord annotations, varying a time lag. A maximum correlation at a lag of zero would indicate good alignment as the audio changes at the same time as the annotation changes. First, the CQT of the audio was computed following the procedure defined in Section 3.1.1.1. The derivative of the CQT in the time dimension was then estimated using `librosa`'s `librosa.feature.delta` function. The chord annotations were converted to a binary vector, where each element corresponds to a frame in the CQT, and is 1 if a chord change occurs at that frame, and 0 otherwise. Both the CQT derivatives and binary vectors were normalised by subtracting the mean and dividing by the standard deviation. Finally, cross-correlation was computed using `numpy`'s `numpy.correlate` function. A typical cross-correlation for a song is shown in Figure 3.1. We can see that the cross-correlation repeats every 20 frames or so.

Listening to the song, we can interpret this as one bar-length, where drum transients will be highly correlated with the chord changes.

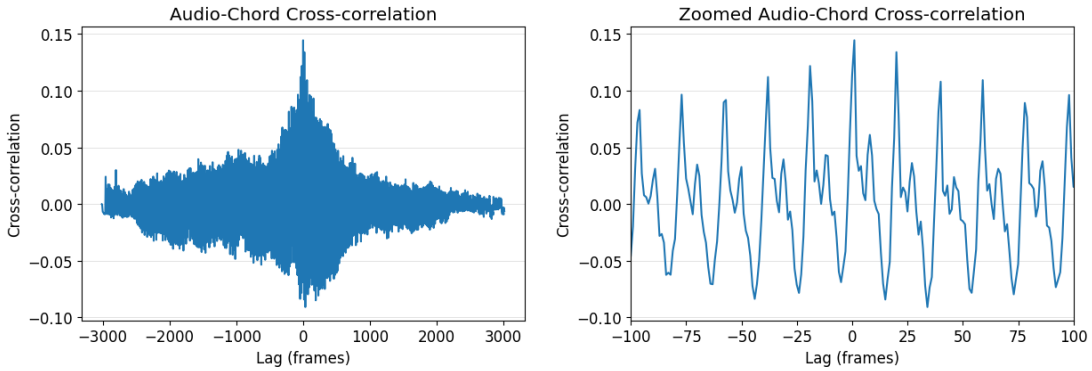

Figure 3.1: Cross-correlation of the derivative of the CQT of the audio and the chord annotations for a single song. We can see correlation peaking in regular intervals of around 20 frames, which corresponds to one bar length in this song. On a larger scale, it is highest on around 0.

To check alignment across the dataset, we can plot the lag of the maximum cross-correlations as a histogram. If we further assume that the annotations are not incorrect by more than approximately 10 seconds, this would mean we could restrict our maximum correlation search to a window of 100 frames either side of 0. A histogram of maximum-lags per song is shown in Figure 3.2 where the maximum is within a window of 100. This reduction does not change the shape of the picture. Rather, focusing on a reduced set of frames allows more detail to be visible. A final simple check was done by looking at the lengths of the audio and chord annotations. A histogram of differences in length is also shown in Figure 3.2. The majority of songs have a maximum lag in the region of 0, with a few outliers. This can be attrubted to noise. Furthermore, the majority of songs have a difference in length of 0, with a few outliers, almost all less than a second. This evidence combined with the qualitative analysis was convincing enough to leave the annotations as they are for training.

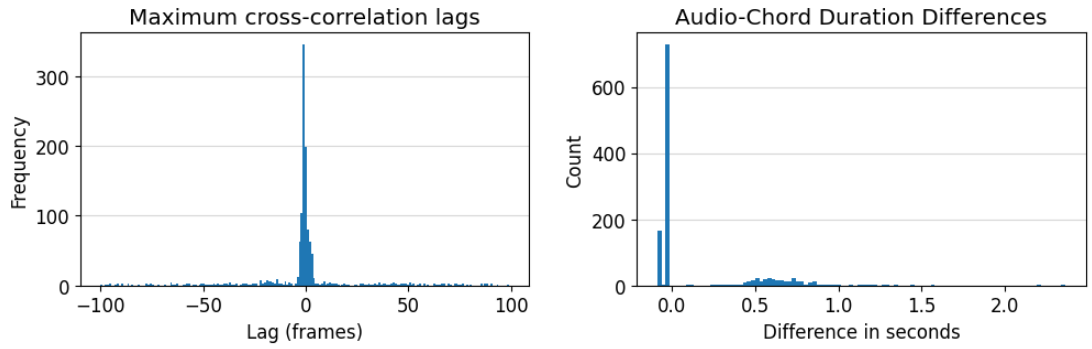

Figure 3.2: Cross-correlation of the derivative of the CQT of the audio and the chord annotations for a single song. The x-axis is the lag in frames and the y-axis is the correlation. The plot repeats every 100 frames, which corresponds to 4 bars.

Incorrect and Subjective Annotations: Throughout manual listening, no obviously

wrong annotations were found. However, looking at songs which the first trained models perform the worst on, 'Lovely Rita' by the Beatles sticks out. The model consistently guessed chords one semitone off, as if it thought the song was in a different key. Upon listening, it became clear that recording's tuning was not in standard A440 and so the song was removed. No other songs were found to have such issues.

Chord annotations are inherently subjective to some extent. Detailed examples in this dataset are given by Humphrey and Bello (2015). They also note that there are several songs in the dataset of questionable relevance to ACR, as the music itself is not well-explained by chord annotations. However, these are kept in for consistency with other works as this dataset is often used in the literature. Some works decide to use the median as opposed to the mean accuracy in their evaluations in order to counteract the effect of such songs on performance (McFee and Bello, 2017).

3.1.2.2 Chord Distribution

Much of the recent literature has focused on the long tail of the chord distribution (Miller et al., 2022; Rowe and Tzanetakis, 2021), using a variety of methods to attempt to address the issue. It is first helpful to understand the distrbution of chords in the datasets, shown in Figure 3.3. The distribution is broken down both by root and by quality, only for the larger chord vocabualary as the smaller vocabualary does not have the same long-tailed distribution. The plots clearly show that the distribution over qualities is highly skewed, with major and minor chords making up the majority of the dataset, and qualities like majorminor and diminished 7th chords having two to three orders of maginute fewer frames. Another helpful display of chord qualities can be found in the work by Jiang et al. (2019). The distribution over roots is far less skewed, although there is unsurprisingly a preference for chords from simpler keys like C, D or E and fewer in C# or F#.

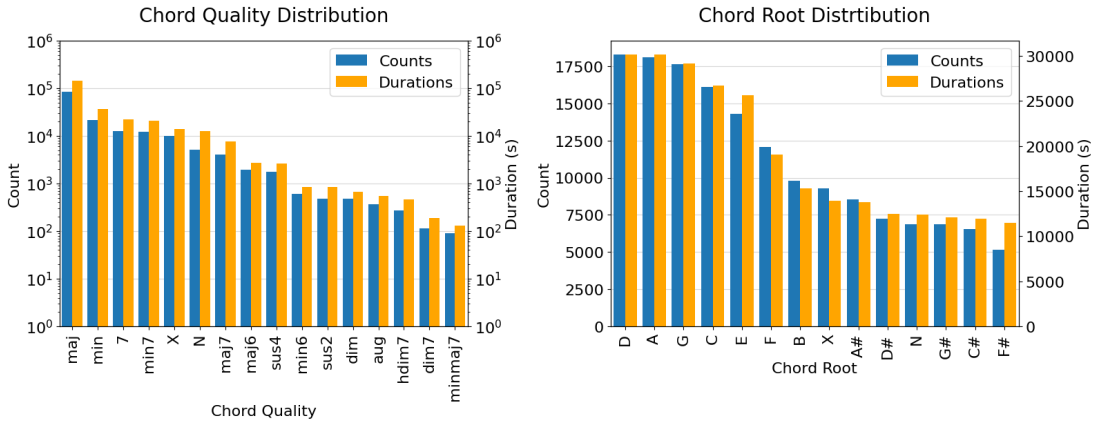

Figure 3.3: Chord distributions in the *Pop* dataset. The plots show both the raw counts in terms of frames and the duration in seconds for each chord root/quality. Note that the scales on the qualities are in log-scale. We observe that the qualities are very imbalanced, with `maj` the most popular, but that roots are relatively balanced.

3.1.3 JAAH Dataset

I was warned by Andrea that the JAAH dataset has not been as commonly used as dataset the Billboard dataset. Therefore he could not guarantee that the audio was aligned for this dataset.

- As yet, JAAH is unused in this work - Data was received as `.flac` files which were first converted to `.mp3` files to be in line with the Billboard dataset - Comparison of the two datasets - Description of the JAAH dataset and its use in this work - Intended to be used as a test set to test the synthetic data generation.

3.2 Evaluation

We evaulated models using the standard `mir_eval` library (Raffel et al., 2014). This library provides a variety of metrics for evaluating the performance of chord recognition models. The metrics used in this work follow standard practice, in using `root`, `third`, `seventh` and `mirex` accuracies. For a given hypothesis and reference chord, these metrics each return 1 if the root/third/seventh/three notes of the chord match and 0 otherwise. They return -1 if the true label is `X` and as such are ignored during evaluation. Finally, we looked at the overall frame-wise accuracy, called `frame`. We choose not to use `majmin`, `triad` or `tetrad` as these are highly correlated with the previous metrics. This makes sense from a theoretical standpoint as the third and seventh are strong indicators of the triad and tetrad of the chord. This was also verified empirically on some initial results, where the `third` and `majmin` accuraciest were always within 1% of one another.

For the above metrics, the mean is computed for each song, and both the mean and median across songs are returned. Although, the mean and median were found to not be very different, the median was consistently slightly higher, likely due to there being more negative outliers than positive outliers. Some of these outliers could be those songs previously identified as being unsuitable for chordal analysis (Humphrey and Bello, 2015). Because of the similarity between the two, we report only the mean throughout this work, chosen as it is more commonly used in recent literature. These metrics can be formally defined as follows:

$$M(f) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{|\{t \in T : y_{i,t} \neq \mathtt{X}\}|} \sum_{t \in T : y_{i,t} \neq \mathtt{X}} f(y_{i,t}, \hat{y}_{i,t})$$

where M is the metric in question, f is a function which calculates the metric for a pair of reference and hypothesis chord labels, defined for each metric as above. N is the number of songs, T is the set of frames in a song, $y_{i,t}$ is the true chord at frame t of song i, and $\hat{y}_{i,t}$ is the predicted chord at frame t of song i.

For some experiments, we looked at three more metrics. The first is the overall frame-wise accuracy, calculated across the entire evaluation dataset rather than on a per-song basis, still ignoring `X` chords, called $\texttt{frame}_{\text{all}}$. The second and third are the mean and median class-wise accuracies, called $\texttt{class}_{\text{mean}}$ and $\texttt{class}_{\text{median}}$ respectively. These

are the mean/median accuracy over each chord class. These metrics are formally defined as follows:

$$\text{frame}_{\text{all}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\{t \in T : y_{i,t} \neq \texttt{X}\}|}\sum_{t\in T: y_{i,t}\neq \texttt{X}} \mathbb{1}_{(y_{i,t}=\hat{y}_{i,t})}$$

$$\text{class}_{\text{mean}} = \frac{1}{C}\sum_{c=1}^{C}\text{acc}(c), \quad \text{class}_{\text{median}} = \text{median}_{c=1}^{C}[\text{acc}(c)]$$

where $\text{acc}(c)$ is the accuracy of class c defined as:

$$\text{acc}(c) = \frac{\sum_{i=1}^{N}\sum_{t\in T: y_{i,t}\neq \texttt{X}} \mathbb{1}_{(y_{i,t}=c \wedge \hat{y}_{i,t}=c)}}{\sum_{i=1}^{N}\sum_{t\in T: y_{i,t}\neq \texttt{X}} \mathbb{1}_{(y_{i,t}=c)}}$$

where C is the number of chord classes, and N, T, $y_{i,t}$ and $\hat{y}_{i,t}$ are as defined before.

This metric is intended to measure the model's performance on the long tail of the chord distribution. It is important to measure both the mean and median as the distribution of accuracies is skewed. These last two metrics are the only ones take into account the entire evaluation dataset at once. Some songs may contain very few chords of a particular chord or quality, making the per-song metrics very noisy.

For the majority of experiments, the metrics on the validation set are used to compare performance. The test set is held out for use only to compare the final performances of selected models.

Finally, other evaluation tools were used, such as confusion matrices and accuracies over particular chord qualities and classes. A small amount of qualitative analysis was also done, looking at the predictions of the model on a few songs. This was done to understand the model's behaviour and to identify any systematic errors.

3.3 Training

Three variants of the dataset were used for training, validation and testing. For training, an epoch consisted of randomly sampling a patch of audio from each song in the training set. The length of this sample was kept as a hyperparameter, set to 28 seconds for the majority of experiments. For testing, the entire song was used was performance was found to be better if the model was allowed to see the entire song at once. Because songs are of variable length, songs were split into patches of the same length as the training patches when validating mid-way through training. This meant that computation could be parallelised without many padded frames. For all variants, frames in the batch were padded to the maximum length of the batch and ignored for loss or metric calculation.

The models were trained using the `PyTorch` library (Paszke et al., 2019). All final experiments were run on the ML Teaching Cluster with a variety of Nvidia GPUs, mostly GTX 1060's or GTX 1080's. Unless state otherwise, models were trained with the Adam optimiser (Kingma and Ba, 2015) with a learning rate of 0.001 and pytorch's

`CosineAnnealingLR` scheduler, set to reduce the learning rate to 1/10th of its initial value over the run. Models were trained to minimise the cross entropy loss between the predicted chord and the true chord distributions. The models were trained with a batch size of 64 for a maximum of 150 epochs unless stated otherwise. This batch size was based on brief experiments testing the speed of an epoch. Validation was conducted every 5 epochs in order to save on computation time. Optionally, training was stopped early if the validation loss did not improve for 25 epochs. The model was saved whenever the validation loss improved. Each training run took approximately 30 minutes of GPU time.

For the majority of experiments, a random 60/20/20% training/validation/test split was used. This contrasts much of the literature which uses a 5-fold cross validation split, as introduced by Humphrey and Bello (2015). We do not maintain this split in order to obtain clean estimators of the generalisation error using the held-out test set. The split was kept constant across experiments. Later, models were re-trained on the combined training and validation sets and tested on the test set. In Chapter 5, the models were trained on the entire *Pop* dataset and tested on the *JAAH* dataset.

Chapter 4

Model Comparison

In this section, we compare various model on the *Pop* dataset. We start by describing a baseline model. We then implemement a CRNN from the literature and investigate its behaviour. We look at weighting the loss function, training on a more structured loss function, and performance of a transformer model. Finally, we investigate the use of generative features in the model and compare the best models on the held-out test set.

4.1 Logistic Baseline

As a simple baseline, we consider a single layer NN which treats each frame independently. Then the layer takes input of size 216 and outputs a V-dimensional vector, where V is the cardinality of the chord vocabulary. The outputs are then passed through a softmax layer to get the probability of each chord and the cross-entropy loss is calculated. We call this model *Logistic*.This can effectively be seen as a logistic regression model trained using SGD. We could have used a logistic regression model implemented in `sklearn` for example which may have found better minima but implementing it as a neural network was fast and easy and unlikely to yield significantly different results.

A grid search on learning rates and learning rate schedulers was conducted on the sets `[0.1, 0.01, 0.001, 0.0001]` and `[Cosine, Plateau, None]` respectively. `Plateau` halves the learning rate when the validation loss hasn't improved for 10 epochs and `Cosine` is as described in Section 3.3. The best model was found to be a learning rate of 0.01 with a `Cosine` scheduler. This best model was chosen for having the highest score on the most of the metrics in validation set. All models with learning rates of 0.01 or 0.001 converged within 150 epochs. Although the best model had learning rate 0.01, a learning rate of 0.001 over 150 epochs had a more stable validation accuracy. The model's results can be seen in Table 4.1. Full results are ommitted as they are not relevant to the main discussion. The model serves simply as a baseline to compare the more complex models to. These results give us the first empirical evidence that the task is non-trivial. The model is only able to predict the root of the chord with a mean frame-wise accuracy of 0.64 and a mirex of 0.65. The model identifies both the root and the third with an accuracy of 0.56 but struggles more with the seventh with an accuracy of 0.44. The lowest scores are on class-wise accuracies able to predict the

class of the chord with $\texttt{class}_{\text{mean}} = 0.13$ and $\texttt{class}_{\text{median}} = 0.03$. This gives us the first insight into each of the evaluation metrics and what we can hope from more complex models and other improvements.

Model	frame	root	third	seventh	mirex	$\text{class}_{\text{mean}}$	$\text{class}_{\text{median}}$
Logistic	0.42	0.64	0.56	0.44	0.65	0.13	0.03

Table 4.1: Baseline model results

4.2 CRNN

4.2.1 Model Description

We implement a Convolutional Recurrent Neural Network (CRNN) as described in McFee and Bello (2017). The model takes as input a matrix of size $I \times F$ where I is the number of input features and F is the number of frames. The model passes the input through a layer of batch normalisation, before being fed through two convolutional layers with ReLU after each one. The first convolutional layer has a 5×5 kernel, and outputs only one channel the same size as the input. It is intended to smooth out noise and spread some information across adjacent frames about sustained notes. The second layer has a kernel of size $1 \times I$, and outputs 36 values. This essentially acts as a linear layer across frames with shared inputs. The output is passed through a bi-directional GRU (Cho et al., 2014), with hidden size initially set to 256 and a final dense layer with the softmax activation. This produces a vector of length V for each frame, where V is the size of the chord vocabulary.

The authors of the model also propose using second GRU as a decoder before the final dense layer, called 'CR2'. However, we believe that a similar effect could be achieved with more layers in the initial GRU. Furthermore, both in the paper and in brief empricial tests of our own, the results with 'CR2' were indistuingishable from the model without it. We therefore do not include it in our final model. Results are ommitted as they are neither relevant nor interesting.

4.2.2 Small to Large Vocabualary

Initial experiments were conducted on the simpler chord vocabualary with $V = 25$. Only if the model could somewhat accurately classify the smaller vocabulary and if performance did not decrease on the smaller chord vocabualary when trained on the larger vocabulary, would we proceed to using the larger vocabualary. In keeping with with the methodology in McFee and Bello (2017), we initially run with a learning rate of 0.001. We reduce the learning rate to half its previous value if the validation loss hasn't improved in 10 epochs and stop training if it has not improved after 25 epochs, with a maximum of 100 epochs. Training samples were set to 10 seconds long. Model convergence was manually checked using the validation and training losses over epochs. A plot of the training history used to ensure check convergence is shown in Figure 4.1.

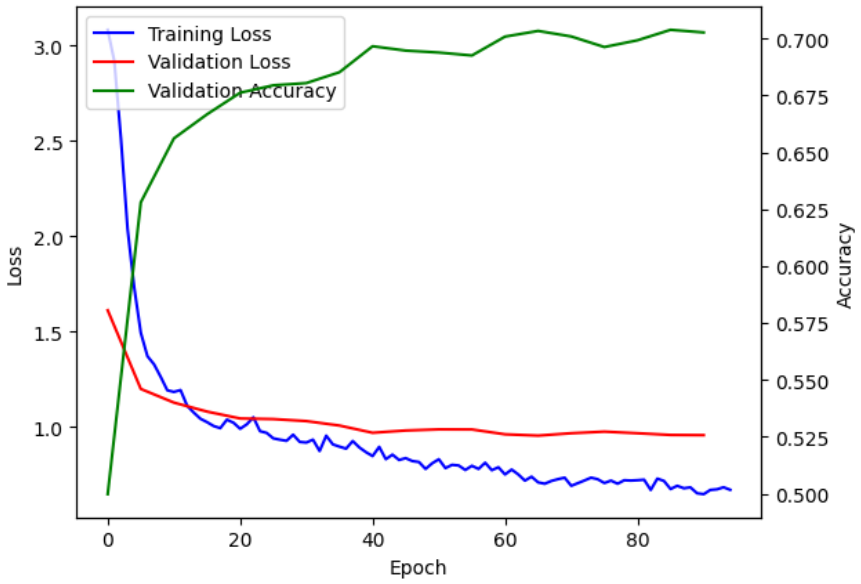

Figure 4.1: CRNN model training history on small vocabualary. We see the training loss and the validation loss and accuracy. The accuracy here is over all chord labels, not ignoring `X` as in final metric calculations. Training was stopped early at epoch 79. We can see the validation loss flattening out after around epoch 50. However, the model could have continued to be trained as it has not started overfitting yet. This behaviour later contributed to the argument for the removal of early stopping.

Results are shown in Table 4.2. For comparison, the table also shows the performance of the same model trained with the large vocabulary, $V = 170$ and its predictions mapped back to the smaller vocabualary. A confusion matrix over chord roots of the model trained on $V = 26$ is shown in Figure 4.2. The model performs better than the baseline on the larger vocabualary, which is to be expected given the nested nature of the models, and the harder task on the larger vocabulary. From the confusion matrix, it becomes clear that many of the mistakes the model is making lie in the `X` symbol, which consitutes just over 7% of the smaller vocabulary dataset. Chords with qualities like `sus4` could be confused with `major` by a reasonable model but are represented with `X` in the smaller vocabulary. Interestingly, the model trained with $V = 170$ performs nearly as well on all metrics as the model trained with $V = 25$. This implies that training with $V = 170$ allows the model to learn almost all the relevant information about the smaller vocabulary, and gives it the chance to learn something about the larger vocabulary as well. Therefore, we proceed with the larger vocabulary for the rest of the experiments.

While some other works continue to measure performance on the smaller vocabualary (Park et al., 2019), we believe more metrics distract from the main goal of increasing performance across a wider range of chords. Additionally, the `third` metric capture much of the information we would look for in the evaluating with $V = 25$. We therefore only measure performance on the larger vocabualary from now on.

Model	V for training	root	third	$\text{class}_{\text{mean}}$	$\text{class}_{\text{median}}$
CRNN	26	0.79	0.77	0.74	0.74
CRNN	170	0.78	0.74	0.72	0.73

Table 4.2: CRNN model results on the small vocabualary with $V = 26$. The other metrics are omitted as they are identical to `third` for classification with $V = 26$.

4.2.3 Hyperparameter Tuning

After progressing to the larger vocabualary, we thought it a good time to conduct some hyperparameter tuning.

4.2.3.1 Learning rates

We perform gridsearch on the same learning rates and learning rate schedulers as for the *Logistic* model. The learning rates were in the set of `[0.1, 0.01, 0.001, 0.0001]` and the learning rate schedulers to `[Cosine, Plateau, None]`. We remove early stopping in order to check for convergence and overfitting without the possibility of a pre-emptive stop. Judging by training graphs seen in 4.3, the best learning rate is 0.001. Any lower and we do not converge fast enough; any higher and gradient updates cause the validation accuracy to be noisy. These figures also show that the validation loss does not get worse after convergence. We conclude that the model is not quick to overfit, perhaps due to the random sampling in the training process. Combined with the fact that training is relatively quick and we only save on improved valiation loss, we decided to remove early stopping. We therefore conduct future experiments without early stopping and with `lr=0.001` and `Cosine` scheduling.

We report a subset of metrics in Table 4.3. The best performing model by validation metrics was found to be with `lr=0.001` and `Cosine` scheduling. However, there were no large differences in performance between the learning rate schedulers. We proceed with these hyperparameters as defaults for the rest of the experiments.

lr	scheduler	root	third	seventh	mirex	$\text{class}_{\text{mean}}$	$\text{class}_{\text{median}}$
0.01	Cosine	0.71	0.68	0.55	0.76	0.13	0.00
0.001	Cosine	**0.77**	**0.73**	**0.60**	0.80	0.17	**0.01**
0.0001	Cosine	0.70	0.65	0.54	0.70	0.10	0.00
0.001	Plateau	0.74	**0.73**	**0.60**	0.80	**0.18**	**0.01**
0.001	None	0.71	0.70	0.58	**0.82**	0.17	0.00

Table 4.3: CRNN model results on the large vocabualary with different learning rates and schedulers. Overall, a learning rate of 0.001 and a scheduler of `Cosine` performs the best in many metrics, though a scheduler of `Plateau` performs just as well or better on many metrics. We prioritise the performance of the model on the root as this is more important than the `mirex` metric.

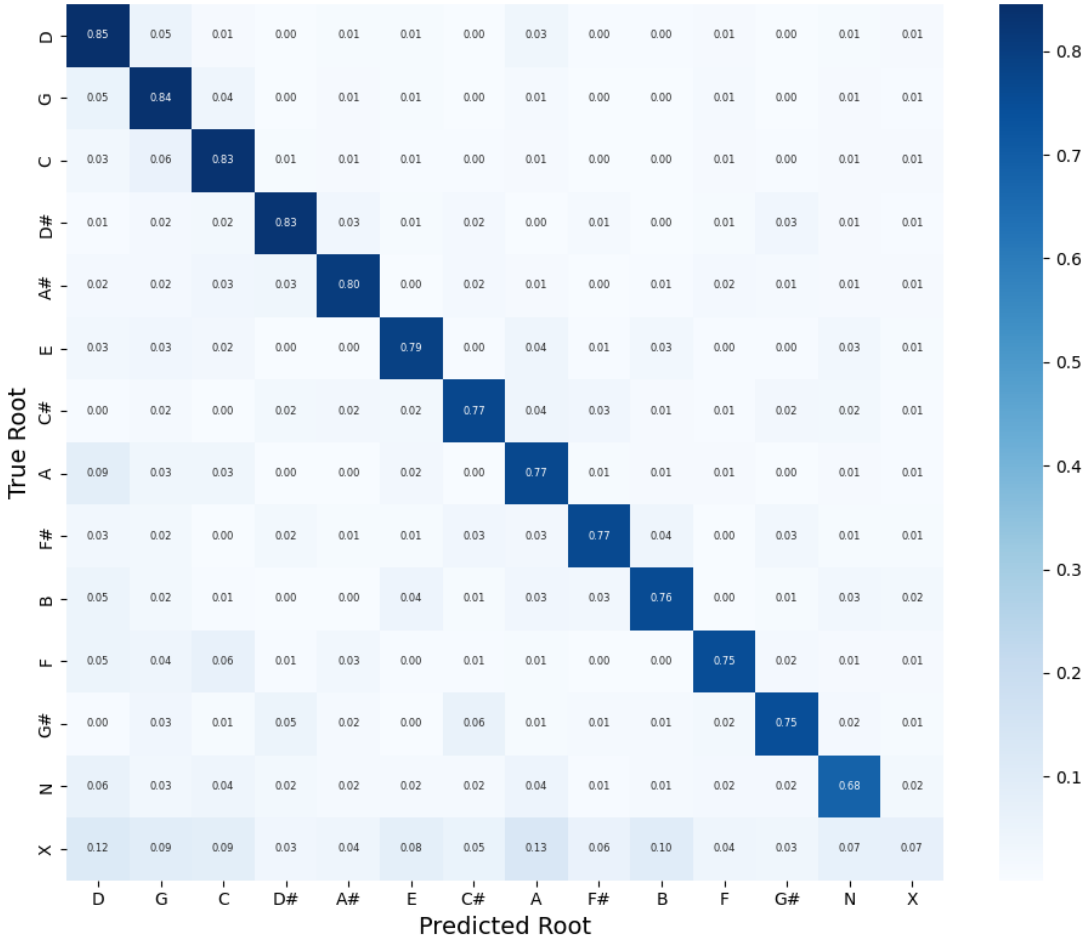

Figure 4.2: Confusion matrix over roots of the CRNN model trained on the small vocabualary. The values have been normalised by their counts, such that the values on the diagonals are recall metrics. There is a clear outlier in the model's recall with the label `X`, at just 0.07. It also performs poorly on `N`.

4.2.3.2 Model Hyperparameters

With this learning rate and learning rate scheduler fixed, we perform a random search on the number of layers in the GRU, the hidden size of the layers in the GRU and the training patch segment length. The search is performed by independently and uniformly randomly sampling 32 points in the sets $\texttt{hidden\_size} \in \{64, 65, \ldots, 512\}$, $\texttt{num\_layers} \in \{1, 2, 3\}$ and $\texttt{segment\_length} \in \{10, 11, \ldots, 60\}$. A sample of the results are shown in Table **??**. The models were then ranked according to each metric and their ranks for each metric added up. The models were ordered by this total rank. The best model was found to have a hidden size of 201, a single layer GRU and a segment length of 28, although the differences between models were relatively small.

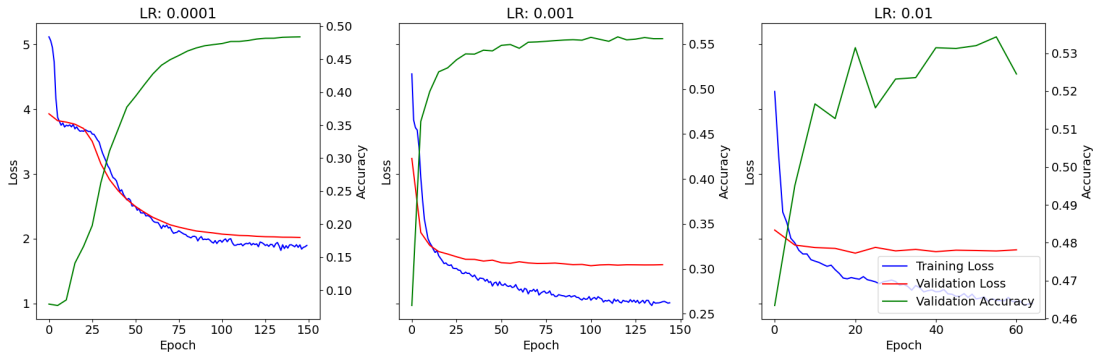

Figure 4.3: Training graphs for the CRNN model with different learning rates. The learning rate of 0.001 seems to be the best, as it converges quickly and does not overfit.

4.2.4 Model Analysis

4.2.4.1 Transition Frames

4.2.4.2 Smoothing

4.2.4.3 Performance Across the Context

4.2.4.4 Qualities and Roots

4.2.5 Long Run SGD

Using SGD in place of Adam as an optimiser has been shown to improve performance in some cases, especially for long-running convergence. We therefore try training the model for 2000 epochs

4.2.6 Hop Length

- Results on different hop lengths. - Metrics aren't directly comparable due to different number of frames.

4.3 Weighted Loss

4.4 Pitch Augmentation

- Two methods: - On CQT [2019] not good - Using MUDA on the audio [everyone else], works?

4.5 Structured Training

- Add structured loss and retrain. From literature. Should (?) further improve accuracies on sevenths etc.

4.6 Transformer

- Make own transformer architecture, similar to BTC or HarmonyTransformer, or Curriculum learning. How to train and evaluate?

4.7 Using Generative Features

- As in [MelodyTranscriptionViaGenerativePreTraining], use Jukebox (?) to generate features at frames (? is it possible to do it on the same frames?). Train on these features and evaluate.

4.8 Results on Test Set

- Directly compare CRNN, weighted loss, pitch augmentation, structured, transformer, generative features on the test set.

Chapter 5

Synthetic Data Generation

5.1 Performance on JAAH

- The performance of existing models on JAAH

5.2 Generation method

- Generation method.

5.3 Experiments

- Brief description of the experiments and metrics I'm looking at

5.4 Results

- Comparison on normal data - Comparison on JAAH

5.5 Qualitative Analysis

- Qualitative analysis of the results

Chapter 6

Conclusions

- What do the results say?

Further work: - More data e.g. HookTheory - Larger vocabulary? Inversion as in? Why didn't I do this? - Make into a useful tool by outputting chords in a more useful format - Investigate other methods of synthetic data generation?

Bibliography

Adam Berenzweig, Beth Logan, Daniel P. Ellis, and Brian Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

Judith Brown. Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89:425–, 01 1991. doi: 10.1121/1.400476.

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

Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis, 2021. URL https://arxiv.org/abs/2111.05011.

Chris Cannam, Craig Landone, and Mark Sandler. Omras2 metadata project. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 309–310, 2009.

Rodrigo Castellon, Chris Donahue, and Percy Liang. Codified audio language modeling learns useful representations for music information retrieval, 2021. URL https://arxiv.org/abs/2107.05677.

Tsung-Ping Chen and Li Su. Harmony transformer: Incorporating chord segmentation into harmony recognition. In *International Society for Music Information Retrieval Conference*, 2019. URL https://api.semanticscholar.org/CorpusID:208334896.

Tsung-Ping Chen and Li Su. Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models. *Trans. Int. Soc. Music. Inf. Retr.*, 4:1–13, 2021. URL https://api.semanticscholar.org/CorpusID:232051159.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

John Thickstun Chris Donahue and Percy Liang. Melody transcription via generative pre-training, 2022. URL https://arxiv.org/abs/2212.01884.

Michael Scott Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In J. Stephen Downie and Remco C. Veltkamp, editors, *ISMIR*, pages 637–642. International Society for Music Information Retrieval, 2010. ISBN 978-90-393-53813. URL http://dblp.uni-trier.de/db/conf/ismir/ismir2010.html#CuthbertA10.

Jacopo de Berardinis, Albert Meroño-Peñuela, Andrea Poltronieri, and Valentina Presutti. Choco: a chord corpus and a data transformation workflow for musical harmony knowledge graphs. *Scientific Data*, 10, 09 2023. doi: 10.1038/s41597-023-02410-w.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. URL https://arxiv.org/abs/2005.00341.

Gabriel Durán and Patricio de la Cuadra. Transcribing Lead Sheet-Like Chord Progressions of Jazz Recordings. *Computer Music Journal*, 44(4):26–42, 12 2020. ISSN 0148-9267. doi: 10.1162/comj_a_00579. URL https://doi.org/10.1162/comj_a_00579.

Takuya Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *International Conference on Mathematics and Computing*, 1999. URL https://api.semanticscholar.org/CorpusID:38716842.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical, and jazz music databases. 01 2002.

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.

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. doi: 10.1109/ICMLA.2012.220.

Eric J. Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *International Society for Music Information Retrieval Conference*, 2015. URL https://api.semanticscholar.org/CorpusID:18774190.

Eric J. Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M. Bittner, and Juan P. Bello. Jams: A json annotated music specification for reproducible mir research. pages 591–596, 2014. 15th International Society for Music Information Retrieval Conference, ISMIR 2014 ; Conference date: 27-10-2014 Through 31-10-2014.

Yun-Ning Hung, Ju-Chiang Wang, Minz Won, and Duc Le. Scaling up music information retrieval training with semi-supervised learning, 2023. URL https://arxiv.org/abs/2310.01353.

Junyan Jiang, K. Chen, Wei Li, and Gus G. Xia. Large-vocabulary chord transcription via chord structure decomposition. In *International Society for Music Information Retrieval Conference*, 2019. URL https://api.semanticscholar.org/CorpusID:208334209.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Nadine Kroher, Helena Cuesta, and Aggelos Pikrakis. Can musicgen create training data for mir tasks?, 2023. URL https://arxiv.org/abs/2311.09094.

Kyogu Lee and Malcolm Slaney. Automatic chord recognition from audio using a hmm with supervised learning. pages 133–137, 01 2006.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, Norbert Gyenge, Roger Dannenberg, Ruibo Liu, Wenhu Chen, Gus Xia, Yemin Shi, Wenhao Huang, Zili Wang, Yike Guo, and Jie Fu. Mert: Acoustic music understanding model with large-scale self-supervised training, 2024. URL https://arxiv.org/abs/2306.00107.

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

Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *International Society for Music Information Retrieval Conference*, 2017. URL https://api.semanticscholar.org/CorpusID:3072806.

Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. pages 18–24, 01 2015. doi: 10.25080/Majora-7b98e3ed-003.

Jeff Miller, Ken O'Hanlon, and Mark B. Sandler. Improving balance in automatic chord recognition with random forests. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 244–248, 2022. doi: 10.23919/EUSIPCO55093.2022.9909558.

Jonggwon Park, Kyoyun Choi, Sungwook Jeon, Dokyun Kim, and Jonghun Park. A bi-directional transformer for musical chord recognition. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pages 620–627, 2019. URL https://archives.ismir.net/ismir2019/paper/000075.pdf.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Johan Pauwels, Ken O'Hanlon, Emilia Gómez, and Mark B. Sandler. 20 years of automatic chord recognition from audio. In *International Society for Music Infor-*

mation Retrieval Conference, 2019. URL https://api.semanticscholar.org/CorpusID:208334309.

Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *International Society for Music Information Retrieval Conference*, 2014. URL https://api.semanticscholar.org/CorpusID:17163281.

Luke O. Rowe and George Tzanetakis. Curriculum learning for imbalanced classification in large vocabulary automatic chord recognition. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, pages 586–593, 2021. URL https://archives.ismir.net/ismir2021/paper/000073.pdf.

Matti P. Ryynänen and Anssi P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008. doi: 10.1162/comj.2008.32.3.72.

Gakusei Sato and Taketo Akama. Annotation-free automatic music transcription with scalable synthetic data and adversarial domain confusion, 2024. URL https://arxiv.org/abs/2312.10402.

Keisuke Toyama, Taketo Akama, Yukara Ikemiya, Yuhta Takida, Wei-Hsiang Liao, and Yuki Mitsufuji. Automatic piano transcription with hierarchical frequency-time transformer, 2023. URL https://arxiv.org/abs/2307.04305.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Avery Wang. An industrial strength audio search algorithm. 01 2003.

Jan Weil, Thomas Sikora, Jean-Louis Durrieu, and Gaël Richard. Automatic generation of lead sheets from polyphonic music signals. pages 603–608, 01 2009.

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. doi: 10.23919/EUSIPCO.2019.8902741.

Appendix A

First appendix

A.1 First section