



NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

**SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**

BSc THESIS

Deep Learning in Audio Chord Estimation

Theofanis A. Aslanidis

Supervisors: **Aggelos Pikrakis**, Assistant Professor, University of Pireaus
Yannis Kopsinis, Libra AI
Panagiotis Stamatopoulos, Assistant Professor, NKUA

ATHENS

JUNE 2020



ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βαθιά Μάθηση για Αναγνώριση Μουσικών Συγχορδιών

Θεοφάνης Α. Ασλανίδης

Επιβλέποντες: Άγγελος Πικράκης, Επίκουρος Καθηγητής, Πανεπιστήμιο Πειραιά
Γιάννης Κοψίνης, Libra AI
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής, ΕΚΠΑ

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2020

BSc THESIS

Deep Learning in Audio Chord Estimation

Theofanis A. Aslanidis

S.N.: 1115201500013

SUPERVISORS: **Aggelos Pikrakis**, Assistant Professor, University of Pireaus
Yannis Kopsinis, Libra AI
Panagiotis Stamatopoulos, Assistant Professor, NKUA

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βαθιά Μάθηση για Αναγνώριση Μουσικών Συγχορδιών

Θεοφάνης Α. Ασλανίδης
A.M.: 1115201500013

Επιβλέποντες: **Άγγελος Πικράκης**, Επίκουρος Καθηγητής, Πανεπιστήμιο Πειραιά
Γιάννης Κοψίνης, Libra AI
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής, ΕΚΠΑ

ABSTRACT

Each music piece consists of a set of different audio chords. These chords are the song's foundation and a skilful musician can identify them by ear. Although, most musicians can identify audio chords, most non musically trained people cannot recognize them. This thesis researches the use of neural networks and their importance, in the process of identifying audio chords. Neural networks have shown great application on identifying objects, as well as on extracting contextual information through time. The combination of those characteristics is what this thesis will explore. More specifically, in this thesis what is going to be presented is the power of a recurrent convolutional neural network in comparison to other architectures for the purpose of identifying objects (chords) that have an association through time.

Subject Area: Deep learning on audio, Automatic chord recognition

Keywords: Neural Networks, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks, R – CNN, Audio Chord Estimation

ΠΕΡΙΛΗΨΗ

Κάθε μουσικό κομμάτι περιέχει ένα συγκεκριμένο αριθμό από διαφορετικές συγχορδίες. Αυτές οι συγχορδίες αποτελούν το σκελετό του τραγουδιού και ένας έμπειρος μουσικός μπορεί να τις αναγνωρίσει με το αυτί. Όμως, ενώ η πλειοψηφία των μουσικών μπορεί να αναγνωρίσει ακουστικά συγχορδίες, οι περισσότεροι μη-μουσικοί που απλά ασχολούνται ερασιτεχνικά, δεν είναι σε θέση να αναγνωρίσουν τις συγχορδίες ακουστικά. Αυτή η εργασία ερευνά την χρήση και σημαντικότητα των νευρωνικών δικτύων στην διαδικασία της αναγνώρισης των συγχορδιών. Τα νευρωνικά δίκτυα, έχουν δείξει μεγάλη δυνατότητα και προοπτική για την αναγνώριση αντικειμένων σε μια ποικιλία δεδομένων, καθώς και αποκωδικοποίηση πληροφορίας σχετιζόμενης άμεσα με το χρόνο. Ο συνδυασμός των χαρακτηριστικών αυτών, είναι αυτό που η εργασία αυτή θα ερευνήσει, πάνω σε μουσικά δεδομένα. Συγκεκριμένα, σε αυτή την εργασία θα παρουσιαστεί η δυνατότητα των αναδρομικών συνελκτικών νευρωνικών δικτύων σε αντίθεση με άλλες αρχιτεκτονικές – πιο απλές – με σκοπό την αναγνώριση αντικειμένων που έχουν εξάρτηση από το χρόνο, όπως οι μουσικές συγχορδίες.

Θεματική περιοχή: Εφαρμογή νευρωνικών δικτύων στη μουσική, Αυτόματη αναγνώριση συγχορδιών

Λέξεις κλειδιά: Νευρωνικά δίκτυα, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks, R – CNN, Audio Chord Estimation

CONTENTS

LIST OF FIGURES	8
LIST OF TABLES	9
1. INTRODUCTION	10
2. BASIC KNOWLEDGE	11
2.1 Music	11
2.1.1 Intervals	11
2.1.2 Chords	12
2.2 Deep Learning	13
2.2.1 CNN	13
2.2.2 RNN	17
2.3 Audio Signal Processing	20
2.3.1 Linear frequency Spectrogram	20
2.3.2 Logarithmic frequency Spectrogram.	21
3. RELATED WORK	22
4. CHORD RECOGNITION PROCESS	24
4.1 Datasets	24
4.2 Pre-Processing	24
4.2.1 DSP	25
4.2.2 Data augmentation	26
4.2.3 Labelling	27
4.2.4 Chord Vocabulary	28
4.2.5 Chord Analysis	29
4.2.6 Slicing	30
4.3 Models – Training	30
4.3.1 Models	30
4.3.2 Training	37
4.3.2.1 Model M1 Experiments	38
4.3.2.2 Model M2 Experiments	41
4.3.3 Imbalanced Classification	44
4.4 Post Processing	47
4.5 Evaluation	48
5. COMPARISON WITH MIREX PAPERS	51
6. FUTURE WORK	52
7. CONCLUSION	53
TABLE OF ACRONYMS	54
REFERENCES	55

LIST OF FIGURES

Figure 1: CNN operation visualization	14
Figure 2: CNN parameters	15
Figure 3: CNN architecture	16
Figure 4: Multiple Input – Multiple Output RNN	17
Figure 5: Multiple Input – Single Output RNN	17
Figure 6: Single Input – Multiple Output RNN	18
Figure 7: Bidirectional RNN	18
Figure 8: LSTM cell	19
Figure 9: ‘Let It Be’ by The Beatles waveform	20
Figure 10: ‘Let It Be’ by The Beatles spectrogram	20
Figure 11: ‘Let It Be’ by The Beatles constant-Q transform	21
Figure 12: ‘Let It Be’ by The Beatles Chromagram	21
Figure 13: SG1 network by Stefan Gasser and Franz Strasser	22
Figure 14: Data flow	24
Figure 15: Constant-Q transform with 192 frequency bins and 24 bins per octave	25
Figure 16: Data augmentation of cat	26
Figure 17: Data augmentation process	26
Figure 18: Chord Vocabulary representation in practice	28
Figure 19: Chord modes	29
Figure 20: Non popular chord appearances	29
Figure 21: Recurrent Model Architecture M1	30
Figure 22: Model M2 Architecture	32
Figure 23: Part 1 – Feature Extraction FE1.	33
Figure 24: Part 2 – Bi-LSTM and Classification	33
Figure 25: Part 1 – Feature Extraction FE2.	34
Figure 26: Part 1 – Feature Extraction FE3.	35
Figure 27: Part 1 – Feature Extraction FE4.	35
Figure 28: Validation Loss Comparison FE3 and FE4	36
Figure 29: Fully Connected Vectorization FCV1	36
Figure 30: Fully Connected Vectorization FCV2	37
Figure 31: Callback History Loss M1	38
Figure 32: Callback History Accuracy M1	38
Figure 33: Root Task Confusion Matrix M1	39
Figure 34: Triad Task Confusion Matrix M1.	39
Figure 35: Fourth Task Confusion Matrix M1	40
Figure 36: Comparison CRNN with RNN	41
Figure 37: Callback History Accuracy FCE3	41
Figure 38: FCE1 Triad Task Confusion Matrix.	42
Figure 39: Left FCE3, Right FCE4 confusion matrices triad task	42
Figure 40: FCE1 Confusion matrix on fourth task	43
Figure 41: Left FCE3, right FCE4 Confusion matrix on fourth task	43
Figure 42: Loss model WL1	45
Figure 43: Loss model WL2	45
Figure 44: All model comparison	46
Figure 45: WL1 fourth task confusion matrix	46
Figure 46: Mirex statistics model comparison bar chart	50
Figure 47: Mirex statistics state of the art model comparison bar chart	51

LIST OF TABLES

Table 1: Intervals	11
Table 2: Triads	12
Table 3: Chord with fourth note	12
Table 4: Jyh-Shing Roger Jang network	23
Table 5: Chord dictionary	28
Table 6: Model description and parameters	31
Table 7: Parameter comparison	35
Table 8: Triad Task model comparison accuracy	42
Table 9: Fourth accuracy model comparison	43
Table 10: Fourth accuracy weighted model comparison	46
Table 11: Mirex evaluation	49

1. INTRODUCTION

Audio chords are a fundamental piece of music and they are built over certain harmonic rules, appearing appealing to the human ear. At the same time, deep learning is widely known for its ability to discover nonlinear relationships on multi-dimensional data. This thesis, using deep learning was aimed to find models that would discover those relationships, as well as the point to that a neural network can learn some fundamental knowledge of music theory.

The problem of audio chord estimation can be found on MIREX (Music Information Retrieval Evaluation eXchange) where multiple scientists are getting involved each year. MIREX provides certain guidelines on datasets, vocabularies, past submissions and evaluation metrics which were used.

Many scientists have used various models and approaches on this matter. This work is quite similar, a combination of the latest approaches using recurrent convolutional networks in large vocabularies. Since each song consists of several chords, means that there are labeled data which can be used for supervised learning.

The project is divided in two discrete parts, the harmonic and the computational. Each one is going to be presented in detail later. Before continuing to the thesis research in detail, it is vital to describe certain things that are taken for granted in the forthcoming chapters.

2. BASIC KNOWLEDGE

This chapter aims to provide all the appropriate basic knowledge that is required for this thesis.

2.1 Music

Here, the reader can find all the necessary information about music, chords and intervals to help him proceed to later chapters of this work.

2.1.1 Intervals

Before jumping into chords, there is a certain term which is the foundation of chords and is called an interval. An interval is a certain distance between 2 notes. They are divided in two categories: Melodic and Harmonic. In melodic intervals the notes are played one after the other, while in harmonic they are played together. In the majority of songs, notes aren't played always together, so this work includes relationships between all kinds of intervals. A set of those intervals consist of a certain chord.

Example,

- The distance between F and G is called a major third, because the distance is 4 semitones.
- The distance between F and C is called a perfect fifth, because the distance is 7 semitones.

Table 1. Intervals

Number of Semitones	Minor/Major/Perfect	Diminished/Augmented
0	Perfect Unison	Diminished Second
1	Minor Second	Augmented Unison
2	Major Second	Diminished Third
3	Minor Third	Augmented Second
4	Major Third	Diminished Fourth
5	Perfect Fourth	Augmented Third
6	-	Diminished Fifth
7	Perfect Fifth	Diminished Sixth
8	Minor Sixth	Augmented Fifth
9	Major Sixth	Diminished Seventh
10	Minor Seventh	Augmented Sixth
11	Major Seventh	Diminished Octave
12	Octave	Augmented Seventh

2.1.2 Audio Chords

The basic chords are called Triads – which contain 3 notes in total. Chords are separated mainly into certain categories. Having an idea about the chord fundamentals is important for understanding the results of this thesis.

- Major
- Minor
- Diminished
- Augmented
- Sustained

Table 2. Triads

Chord	Symbol (on C)	Qualities
Major Triad	C	P1,M3,P5
Minor Triad	Cmin	P1,m3,P5
Augmented Triad	Caug	P1,M3,A5
Diminished Triad	Cdim	P1,m3,D5

Sustained chords on the other hand, are when a part of the triad is replaced by a 2nd or a 4th in turn sus2 and sus4 appear. Example is a Csus4 where it consists of C + F + G.

On top of those triads, certain extra notes can be added, which lead to more complex chords. Such chord as major7, minor7, dominant7, maj6, min6 which are in detail in the table below.

Table 3. Chord with fourth notes

Chord	Symbol (on C)	Qualities
Major Seventh Chord	Cmaj7	P1,M3,P5,M7
Minor Seventh Chord	Cmin7	P1,m3,P5,m7
Dominant Seventh Chord	C7	P1,M3,P5,m7
Augmented Seventh Chord	Caug7	P1,M3,A5,m7
Diminished Seventh Chord	Cdim7	P1,m3,d5,d7
Half Diminished Seventh Chord	Chdim7	P1,m3,d5,m7
Minor-Major Seventh Chord	Cminmaj7	P1,m3,P5,M7
Major Sixth Chord	Cmaj6	P1,M3,M6
Minor Sixth Chord	Cmin6	P1,m3,M6

Chords can appear on a variety of different styles. These styles have to do with the bass of the chord and they are called Inversions. Notation-wise inversions can be either the 3rd, the 5th or (if exists) an extra note – a 7th for instance.

2.2 Deep Learning

Deep learning is a subset of machine learning. It has to do with finding nonlinear relationships in great amounts of data using layers. Different architectures exist in this field, some of which are used in this thesis. Next there is a brief summary to elaborate on how those architectures that were used work.

2.2.1 Convolutional Neural Networks (CNN)

Convolutional network architectures took their name from the mathematical operation of convolution, but what they do is to implement cross-correlation and name it convolution.[1] They are widely known for their use in application that deal with grid-like data, such as images.

As it is widely seen in basic Multi-Layer Perceptron networks, the neuron operation that is their foundation is

$$a_i = x^T w_i + b_i$$

In this technique the weight matrix – which contains the parameters that are tuned – is multiplied with the input x and by adding a bias the outcome a_i is produced, which is the output of the neuron. In MLP, each neuron has its own weight matrix, with dimensions similar to the input x of the neuron.

On the other hand, convolutional neural networks work differently as long as the weights are considered. In this case, the weights 'w' are now called filters, and they are independent from the input. In fact, in terms of size they are very small, allowing to be multiplied by the input, multiple times at different points. In turn, they have a great application on data with grid-like topologies, such as images and videos due to various reasons that are in detail below. Convolutional networks can be found mostly in computer vision which is one of the most highly developing areas of deep learning.

Below, there is an input array and a filter (kernel) who is going to repeat an overlapping multiplication with all different parts of the input and create a feature map. Each iteration of the multiplication of the filter with the input, provide a single number for the output.

$$\begin{bmatrix} 1 & 2 & 5 & 1 & 0 & 0 \\ 4 & 2 & 2 & 4 & 1 & 5 \\ 4 & 0 & 1 & 9 & 6 & 9 \\ 8 & 3 & 5 & 5 & 1 & 3 \\ 9 & 4 & 1 & 1 & 1 & 0 \\ 3 & 0 & 9 & 8 & 2 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -10 & 1 & 0 \\ 8 & -13 & 0 & 1 \\ 14 & -8 & -1 & 3 \\ 5 & -7 & 11 & 7 \end{bmatrix}$$

Which is analysed as shown below

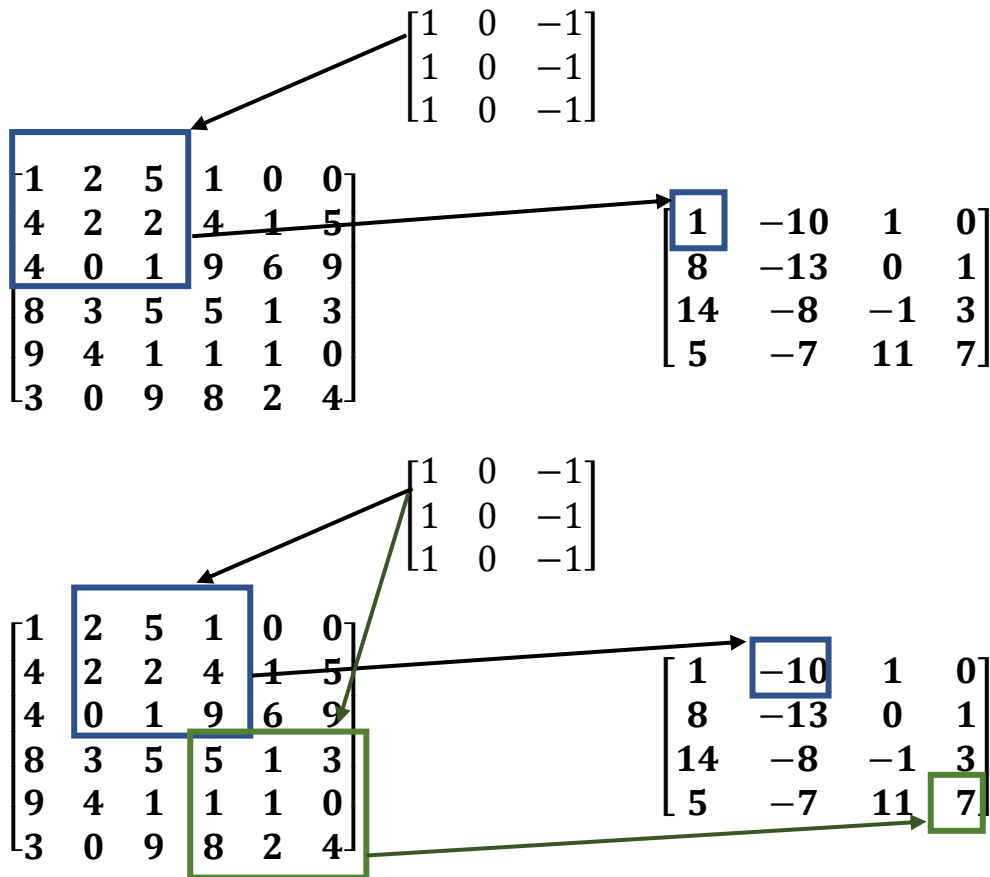


Figure 1. CNN operation visualization

The operation is repeated for every combination inside the input array.

It is evident that convolutional networks overpowered the fully connected networks for these grid-like topology data for some specific reasons. They are more favourable mostly due to the easiness of training and the small number of parameters they require relatively to fully connected ones.

As an example, consider an input image of size 32×32 with 3 filters for red – green – blue.

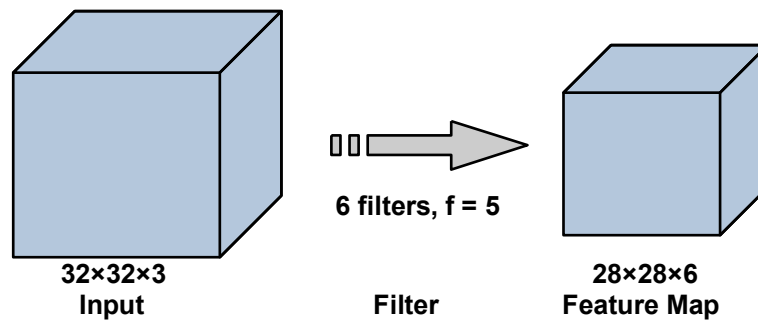


Figure 2. CNN parameters

Comparison between MLP – fully connected layer and a convolutional layer.

- Fully Connected Layer:
 $3.072 \times 4.704 = \mathbf{14M}$ parameters
- Convolution Layer:
 Since $f = 5$ and there are 6 filters
 $5 \times 5 = 25 + 1$ (for bias) = 26×6 (filters) = $\mathbf{156}$ parameters

Useful properties of Convolutions: [2]

- *Parameter sharing*: The importance of parameter sharing, is that the parameters – weights must be useful for all the input data, and not just for a local feature. More specifically, it deteriorates the idea that an image's statistical features are stationary. If an image of a cat, contains a cat on the upper right corner, that convolutional network, can label an image as a cat, even though the cat might be on the lower left corner.
- *Local connectivity*: Restricts the layer from learning local features, and thus provides a useful generalization exploiting the large number of parameters.
- *Spatial Layout*: the spatial layout of the convolution's output, represents the spatial layout of the convolution's input.

Some other building boxes of convolutional networks are the terms pooling, padding and strides.

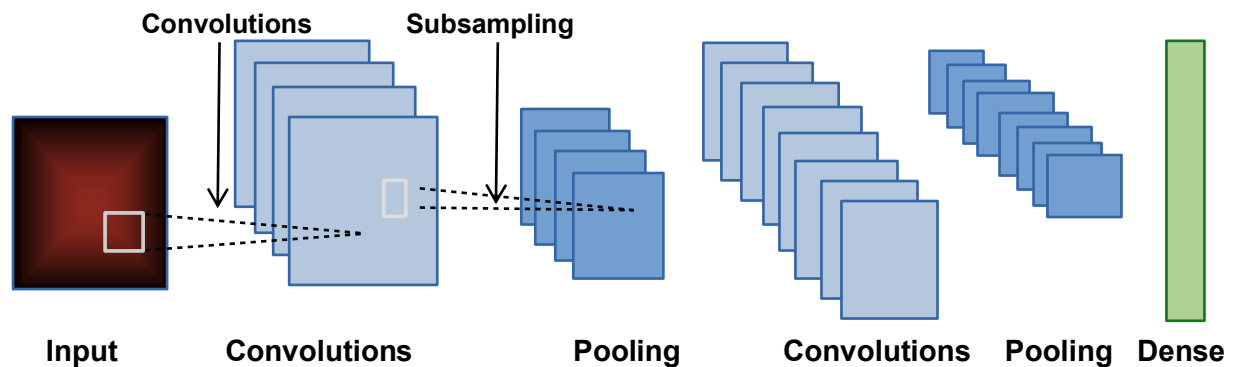


Figure 3. CNN architecture

Pooling is a tool that all convolutional networks use in order to summarize local statistical features, and greatly reduce the output space of a layer, and thus the next layer will have k fewer input features. As is described in [1] this layer can greatly improve the statistical efficiency of the net.

What a pooling layer does, is to summarize the output of the convolutional layer and replace it with statistical summary of neighbourhoods. In applications it can be used either as:

- Max
- Average
- L_2 norm
- Weighted Average

2.2.2 Recurrent Neural Networks (RNN)

Such as convolutional neural networks have shown great use in grid like data, recurrent networks aim to process sequences of values $x^{(1)}, \dots, x^{(T)}$ and find correlations between the data that are close in the sequence. The amazing work of recurrent neural networks is that they can find contextual information through time. There is a certain window in which they can remember information between different time steps, and that depends on a set of things – The architectural structure and the training algorithm [10]. For example, the sentence “During my time in university, I’ve read a lot’s of books” is the same with “I’ve read a lot’s of books, during my time in university”, but a multilayer network can not identify them as the same, unless it is provided with both sequences to train. On the other hand, RNNs have the generalization capability of transferring information in a certain window of time steps. It is an architecture based on unfolding computational graphs. With that in mind, it is understood why those networks are able to flow the information forward and backward in time. Also, exactly as in convolutional networks, they are equipped with the parameter sharing idea. [1]

Recurrent neural networks are divided in different categories, based on the number of inputs and outputs.

List of categories:

1. **Multiple Input – Multiple Output** where the sequence input T_x is equal to length to the sequence output T_y ($T_y = T_x$)

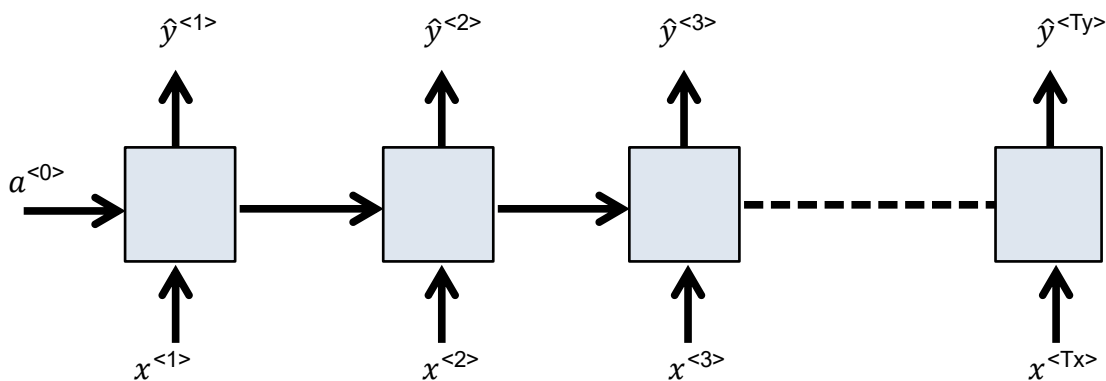


Figure 4: Multiple Input – Multiple Output RNN

In case $T_x \neq T_y$, then a different approach is needed, called encoder – decoder, such as used in machine translation applications, or text generation.

2. **Multiple Input – Single Output**, where the sequence input T_x corresponds to a single output y . Example of this architecture is the case of **Sentiment Classification**.

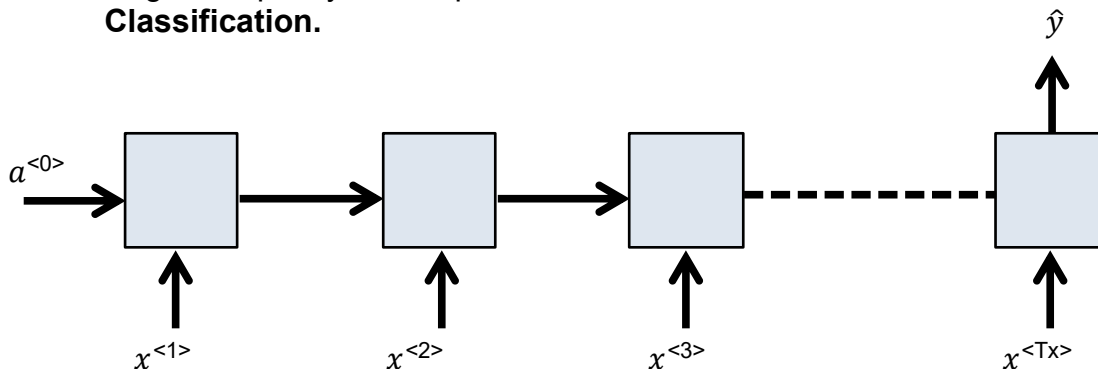


Figure 5: Multiple Input – Single Output RNN

3. Single Input – Multiple Output, where a single input can lead to multiple outputs, such as **Music Generation**.

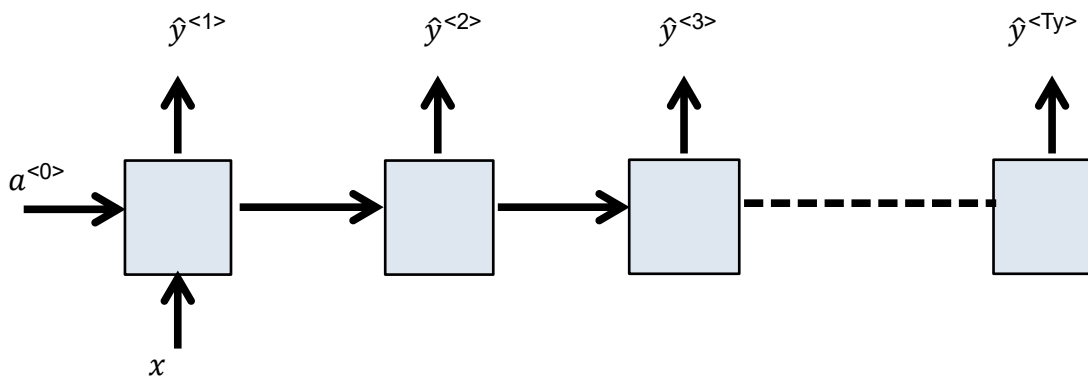


Figure 6: Single Input – Single Output RNN

More specifically, RNNs has shown great results on modelling long term dependencies between acoustic events [6]. More about RNNs can be found on the deep learning book. [1]

In this thesis, as described on later pages, the technique used is the first one – many to many – since there is a label for every time step of input X with a chord Y.

2.2.3 Bidirectional – RNN

In simple RNNs, when predicting for the time step T_K it uses all the information ranging for T_1 all the way up to T_K . However, in most cases, information that appears after T_K and onwards, might be beneficial in the prediction of T_K . In order to overcome those limitations and make good use of the future time steps, no matter how large time data points exist in the sequence, a new architecture was proposed (Mike Schuster and Kuldip K. Paliwal 1997) [10] called Bidirectional Recurrent neural networks that is trained using all available information on both past and future for each time step.

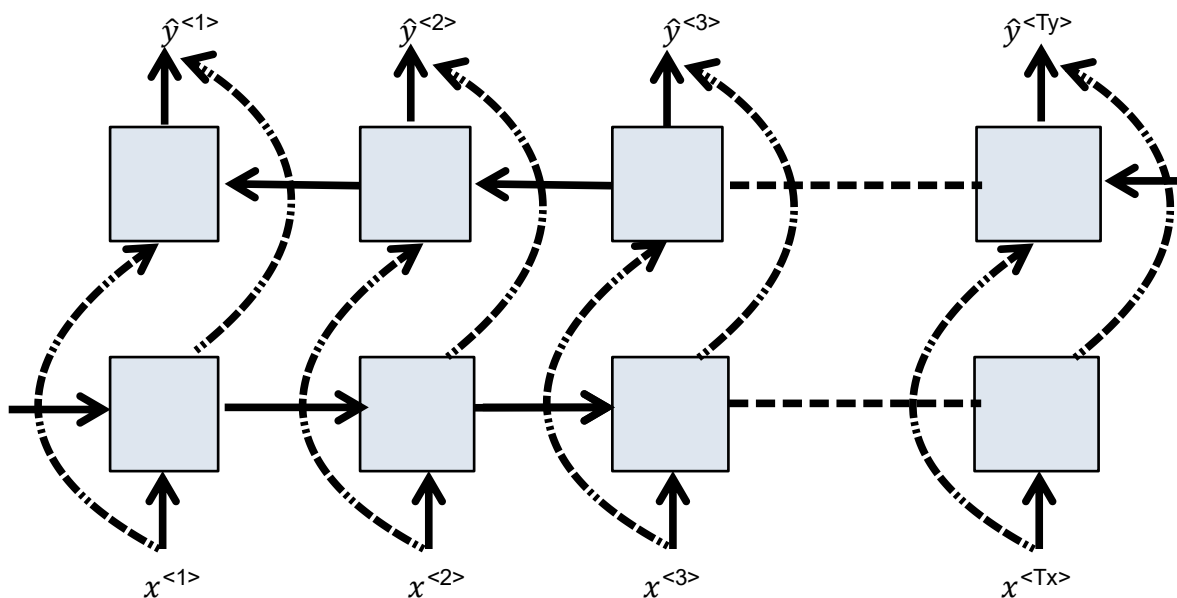


Figure 7. Bidirectional – RNN

2.2.4 LSTM

With conventional RNN techniques, a certain problem is observed for certain training procedures. When provided with large time data the back propagated error quickly vanishes or blows up. On this problem, a major role is played by the magnitude of the weights, which through back propagation, can change the error drastically. Long Short-Term Memory (LSTM) belongs to the family of recurrent neural networks but it is not affected by this problem.[8] With this architecture long time complex tasks could be finally solved. [9]

The foundation of LSTMs is an LSTM cell, which consists of certain gates that are not observed in other recurrent network techniques. Below there is a diagram of the LSTM cell.

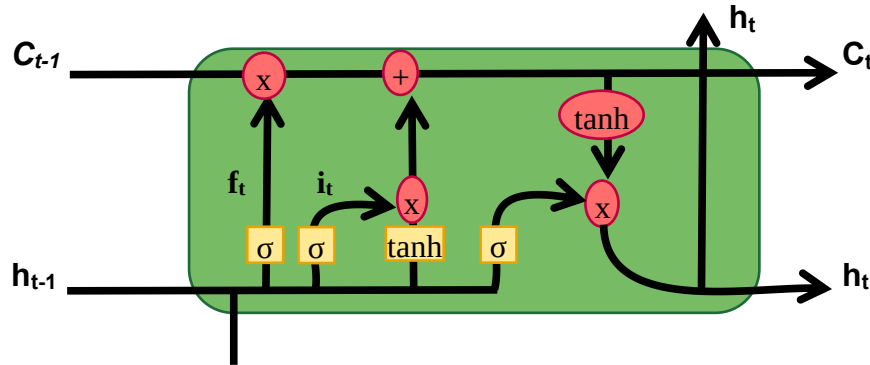


Figure 8. Image of LSTM cell

An LSTM cell takes as input the input data X , the last cell's output and the last cell's state. The C_t is a symbol for the cell state / cell memory

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t)$$

The symbol h_t stands for the cell's output and is calculated by the tanh activation of the cell's memory multiplied with the output gate.

$$h_t = \tanh(C_t) * o_t$$

The update gate or candidate is symbolized with \tilde{C} and calculated as below.

$$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g)$$

In contrast with simple RNN techniques, LSTMs possess – in addition to the outer loop – a linear self-loop inside the LSTM cell whose weight is controlled by the forget gate. This forget gate has values between 0 and 1 since it is a sigmoid output.

$$f_t = \sigma(x_t U^f + h_{t-1} W^f)$$

The external input gate is similar to the forget gate.

$$i_t = \sigma(x_t U^i + h_{t-1} W^i)$$

The output gate of the cell, is also a sigmoid output which can shut off the cell's output h_t .

$$o_t = \sigma(x_t U^o + h_{t-1} W^o)$$

2.3 Audio Signal Processing

Besides from the fundamentals of chords and deep learning, some information about audio signal processing is also necessary. Deep learning algorithms need representations that can provide highly suitable data, in a format that is efficient of the architecture used. In turn, providing raw audio directly to the algorithms is not going to work. Thus, the need for certain digital signal processing algorithms is crucial in order to turn the data in appropriate representations.

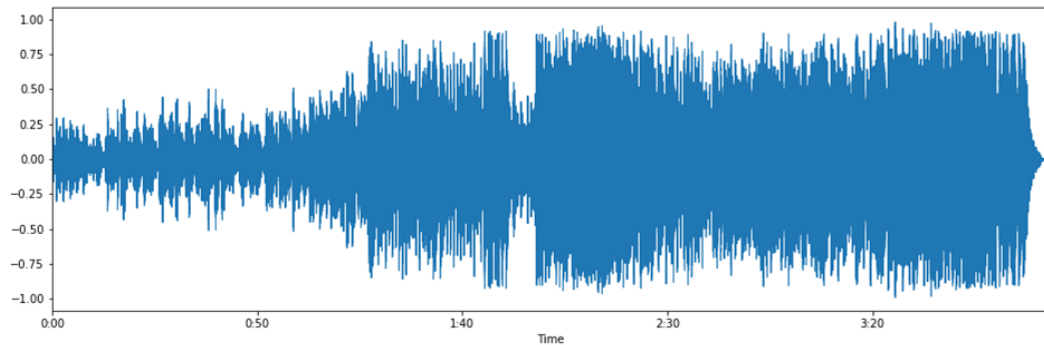


Figure 9. 'Let It Be' waveform

2.3.1 Spectrogram

A spectrogram is a visual representation of a signal. It is a time series of frequency strength, ranging from low to high frequency. It is formed using the short time Fourier transform – stft. The human ear can listen from 20Hz to 20.000 Hz, although by age this window is shrinking.

It can be observed as a two-dimensional graph, with a third dimension represented from colours, in the x-axis there is time, on y-axis frequencies and the colour which represents the strength of the signal in each frequency bin.

By using these spectrograms, sound is shaped into an image. However, there is a major difference between images, and spectrograms in terms of representation because a spectrogram is also a time series of data, a sequence of frequencies that make sense only as a whole.

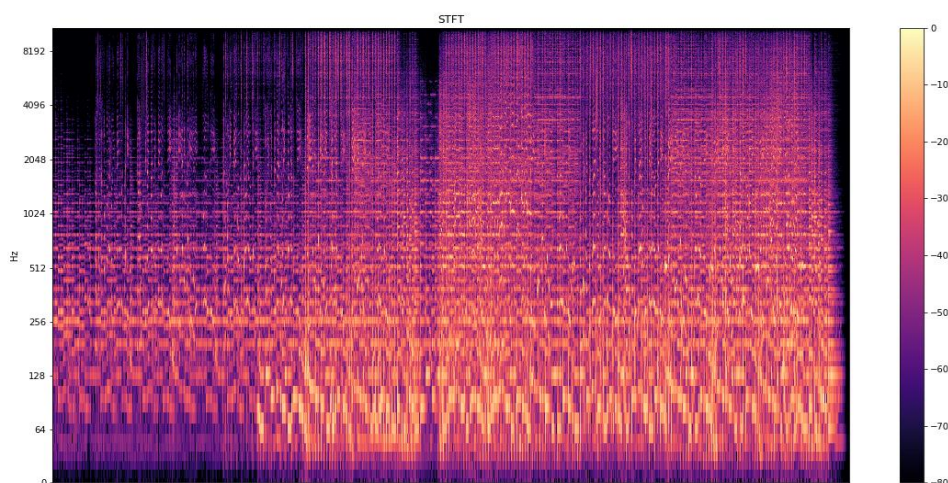


Figure 10. Spectrogram of "Let It Be" by "The Beatles" with hop length = 2048

Hop length is the number of samples between successive audio frames [15]

2.3.2 Logarithmic frequency Spectrogram

This transform is called also a constant-Q transform and it is quite similar to stft, but it differentiates in the frequency spectrum. In constant Q transform there are K-bins of frequencies with a window of W_i . It is a time series of filters logarithmically spaced in frequency. Each filter's window is a multiple of the previous one.

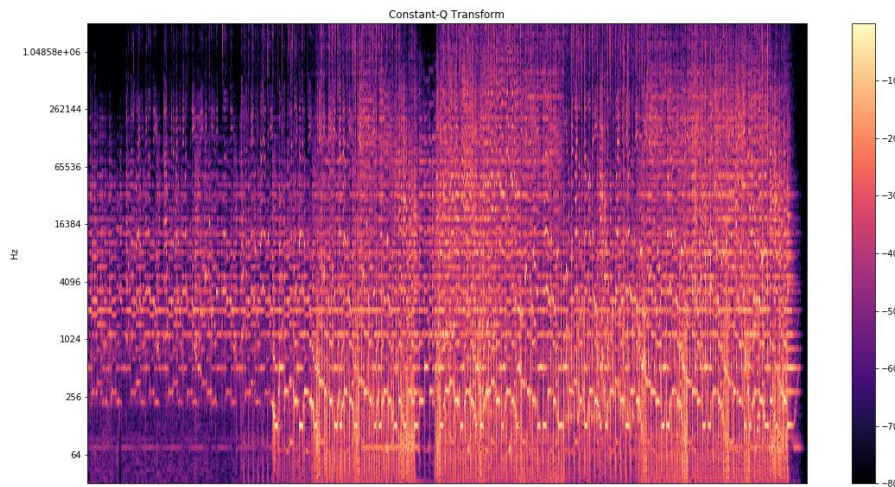


Figure 11. Spectrogram of “Let It Be” by “The Beatles” with #bins = 192, bins per octave = 24 and hop length = 2048

2.3.3 Chromagram

Chromagram is a spectral representation of a signal, that consists of chroma features. Those features are a projection of the musical notes onto 12 bins. Since the distribution of frequencies on notes is known, they can be represented in chroma. A 12-bin spectral representation of 12 bins – 1 bin for each semitone.

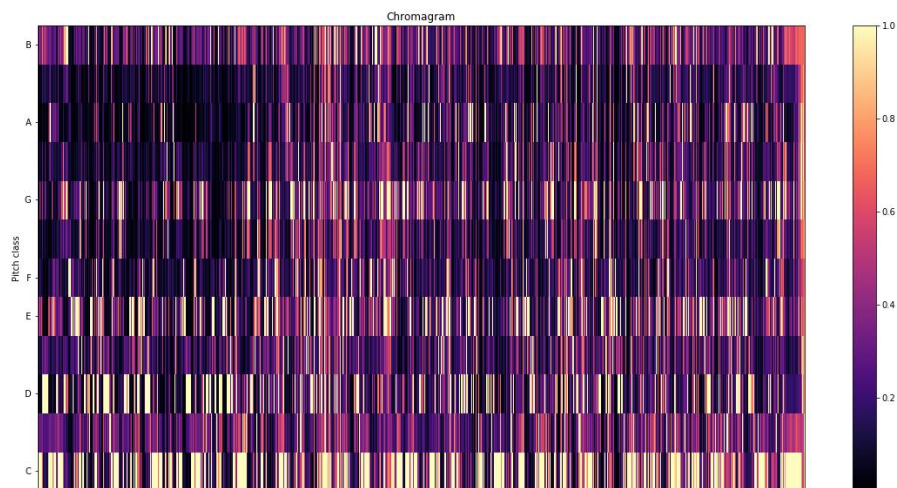


Figure 12. ‘Let it Be’ Chromagram, hop length=2048

3. RELATED WORK

Every year, MIReX holds competitions for various audio specific tasks, amongst them audio chord estimation. Some scientists each year, are presenting models with very good results, on an amount of data that have an incremental trajectory. The most commonly used datasets as of 2019, are Isophonics [1] a dataset that consists of 180 songs by the Beatles, created by Christopher Harte, McGill Billboard [2] created by researchers at McGill University, RWC [3], Robbin Williams [4] and 2002pop[5], as they were listed by Humphrey and Bello.

For audio extraction most scientists use NNLS chroma plugin from Center for Digital Music at the Queen Mary University of London, and train on the entirety of the songs. It uses a Gaussian-HMM as a decoder/encoder in order to extract chromagram features using a probabilistic model.

Model wise, in 2016, Junqi Deng and Yu-Kwong Kwok developed both a deep belief and a BLSTM-RNN network, evaluating their performances on the task. At the same year, Filip Korzeniowski and Gerhard Widmer used a Convolutional neural network followed by a Conditional Random Field.

Later, in 2018, Stefan Gasser and Franz Strasser developed a Convolutional neural network.

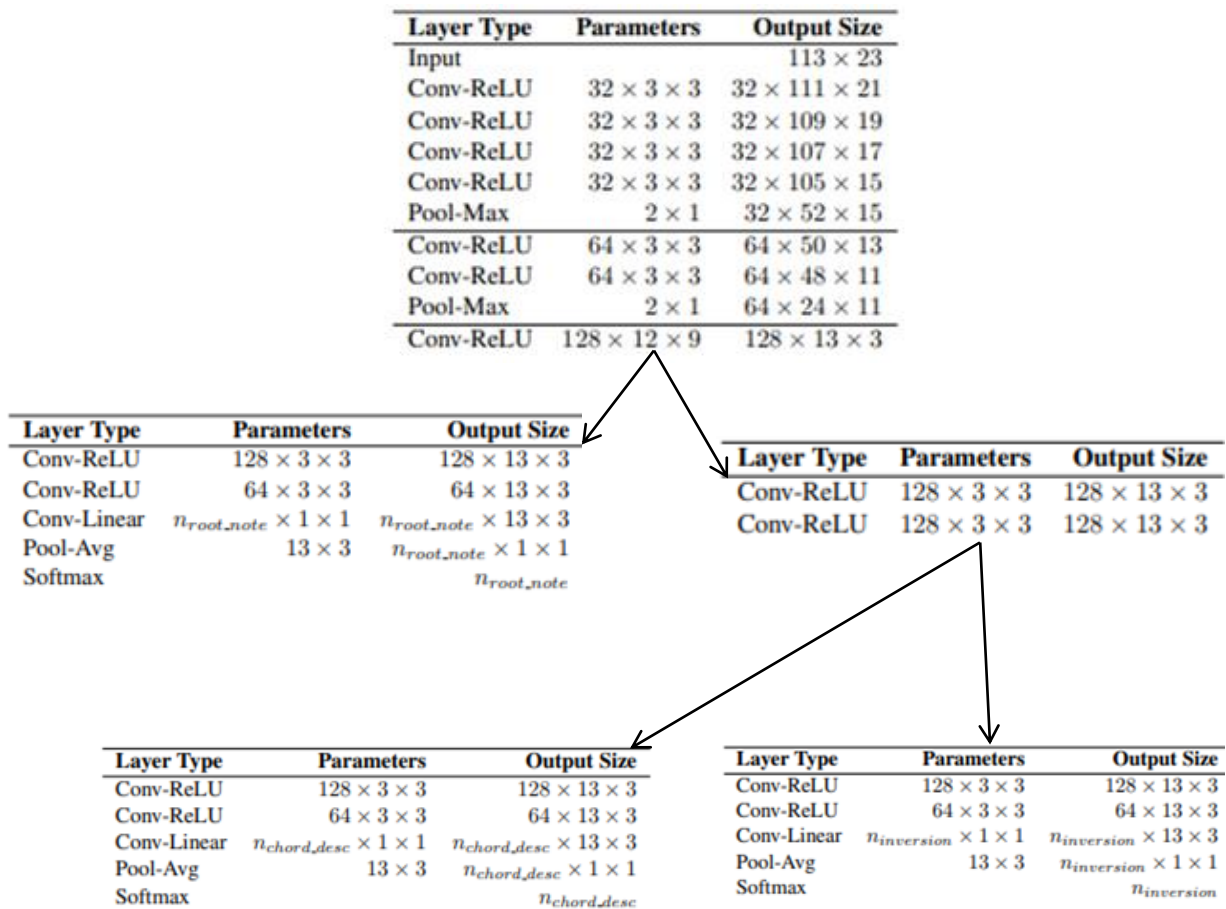


Figure 13: SG1 network by Stefan Gasser and Franz Strasser

It is comprised of different classification tasks, as it is separated in 5 parts. The main part, the classification task for root note, a mutual part of a 2-layer Convolution, followed by the 2 remaining classification tasks – chord description/quality and inversion.

It is important to state the fact that they performed batch normalization after each convolution layer – something that will be seen in this thesis implementation too.

Lastly, in 2018, Jyh-Shing Roger Jang proposed a recurrent convolutional network, followed by a structural chord representation.

Table 4. Jyh-Shing Roger Jang network parameters

Layer Type	Parameters
Convolution	16×3×3
Convolution	16×3×3
Convolution	16×3×3
Max Pooling	3×3
Convolution	32×3×3
Convolution	32×3×3
Convolution	32×3×3
Max Pooling	3×3
Convolution	64×3×3
Convolution	64×3×3
Max Pooling	3×4
Bi-LSTM	128×2
Fully Connected	145

As it is seen from the Mirex 2018 audio estimation results, the latter model of Jang's with the recurrent convolutional network, is more powerful than just the convolutional.

In turn, in this thesis the research will focus on the application of simple recurrent networks in the audio chord classification task in comparison with various versions of a recurrent convolutional network.

4. CHORD RECOGNITION PROCESS

During this thesis, the research was focused on various methods for each step of the process, including pre-processing, model architectures, training hyper parameters and post processing techniques. There are some methods which helped improve the accuracy substantially, and others that didn't make a great impact on the result.

4.1 Datasets

For the training and testing, the dataset used is the Isophonics Beatles, created by Harte [3] providing 180 songs by The Beatles. Because of copyright issues it was difficult to acquire the same audio files that the transcription were made on. Instead, taking the remastered versions of the songs, and using a modification made on the MATLAB scripts that Harte provides, to shift the transcriptions onto the new events on my audio tracks, can lead to perfectly aligned dataset. The format of the data set is 180 audio tracks on mono .wav format at 44kHz sample rate, with their respective chord label files that have the start and the end time of each chord appearance.

4.2 Pre – Processing

When working with artificial neural networks, the input data in most cases is not used in their raw form, in order to provide a solid basis for the algorithm to train. Certain processing and modifications have to be applied on the data, to transform them into appropriate representations. As an example, in this case, a network cannot perform well when seeing input information as waveforms, instead it can perform very well when it sees the corresponding frequency representation of the waveform, called a spectrogram – as described in chapter 2. Also, a network cannot learn label names as characters such as 'Cat', 'Dog', 'Horse'. Upon classifying images of animals, each category has to inherit an encoding, called one-hot encoding which is a vector of size equal to the number of categories full of zeros, except a one placed on the column of each category. Example, 3 categories 'Cat', 'Dog', 'Horse'. The one hot for each category will be respectively: [1 0 0], [0 1 0], [0 0 1].

With all the above in mind, it is time to elaborate on the data that this thesis includes. Each neural network, in order to work it needs some input data called X and the output data called Y. In this case, X data are the track features, and the Y data the chord labels. The purpose of this network is by seeing the track features to be able to identify at each time frame which chord appears. In order to proceed with that mindset, it is vital to do certain processing on the raw data provided by the datasets presented in detail on chapter 4.1.

Below there is a chart of how the data flows until reaching the neural network. Each box is a part in this chapter and will be elaborated on in detail.

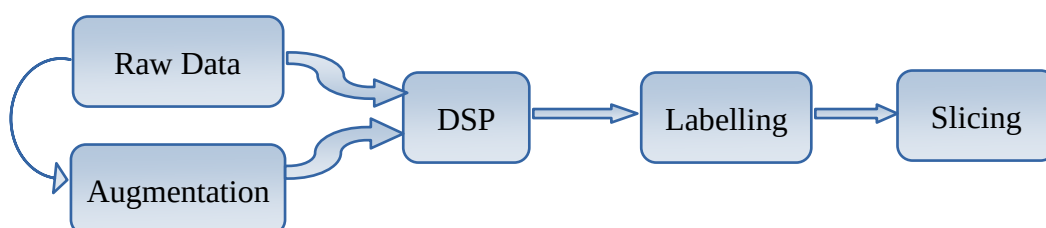


Figure 14. Data flow

4.2.1 DSP

In order to transform the data into a format that can be easily understood and provide a solid ground for feature extraction for the network, the raw audio data were transformed into the related spectrograms.

Three different approaches on the spectrograms:

1. Short time Fourier transform (STFT)
2. Chromagram (1 bin for each semitone)
3. Logarithmic frequency spectrogram (constant-Q)

For each method, the appropriate tests followed and concluded that the best method is the constant-Q transform. Regular linear STFT provided too many frequency bands, and in turn the network had a huge amount of input features, leading to an enormous number of parameters. Chromagram, on the other hand, with 12 input features provided lots of information about the notes used, but little of how they were distributed throughout the spectrum – C2 and C4 are the same notes but provide different information. Also discovering inversions was harder, due to the same reason.

Thus, by using the constant-Q transform it produces an optimal number of features that fit to the later models. Also, constant-Q provided features with the appropriate amount of frequency overlapping, that was useful for the algorithm later to identify features on different frequency bands. Also, the sample rate used was 22050, with 192 number of bins, 24 bins per octave and a hop length of 2048.

After creating a spectrogram for each track, the result was a NumPy array with the input data for each track.

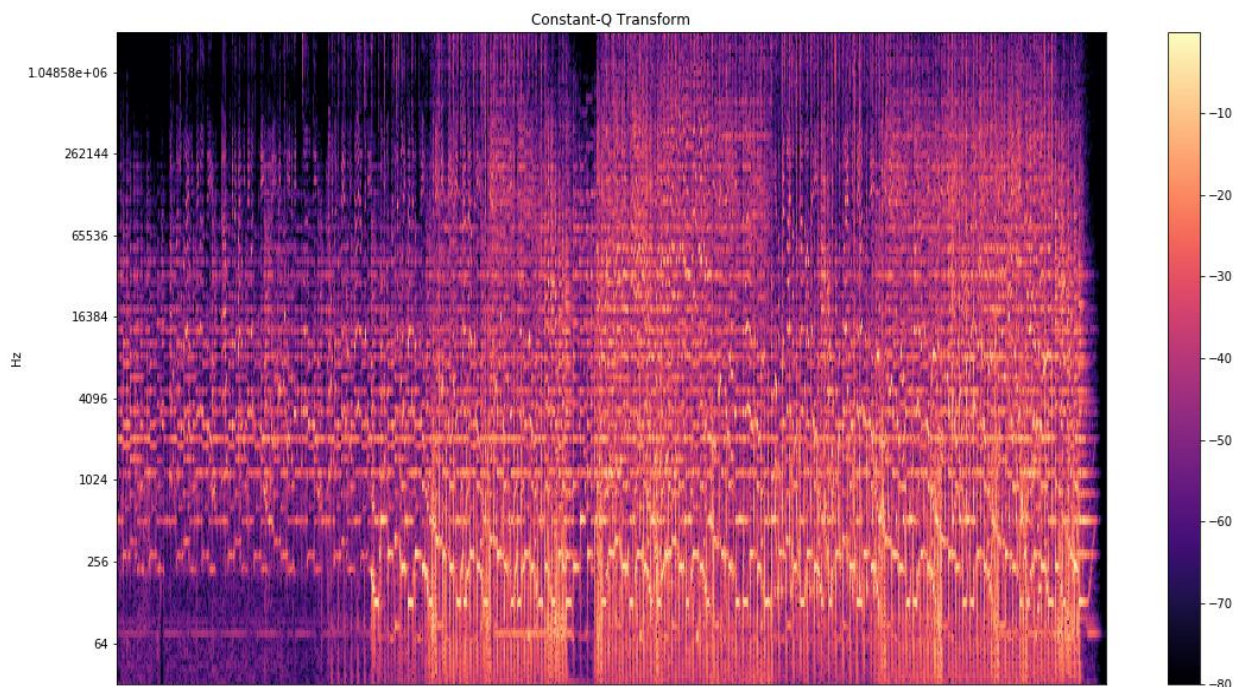


Figure 15. Constant Q Transform with 192 frequency bins and 24 bins per octave

4.2.2 Data augmentation

In computer vision, there is a very important step of the data pre-processing which targets on generalization of the training data. As an example, if a neural network always sees a picture of a cat zoomed in high quality and always upright, if it ever sees an image of a cat in low quality and another position from the ones seen, it may have some trouble classifying it as a cat sometimes. In order to have a generalized neural network, a very good practice is called data augmentation.



Figure 16. Cat example data augmentation

Exactly as mentioned with the images of a cat, the same has application on a picture of a chord. When the neural network sees for every song the relevant spectrograms, it labels them as time series with the appropriate root and quality. The song “Let it be” is mostly a piano with a vocal and corresponds to certain chord labels. If the network, ever sees a piano with a vocal, but playing different chords than those of “Let it Be” there is a possibility that the loss is going to be higher than the training example. That can easily be overturned simply by pitch shifting our existing audio clips certain semitones up and down, generalizing our dataset for every combination of chord that may come for testing to identify.

Also, the addition of Gaussian noise before pitch shifting each song is scientifically proven that reduces overfitting [22]. Adding noise provides a smoother and easier to learn dataset leading to better generalization.

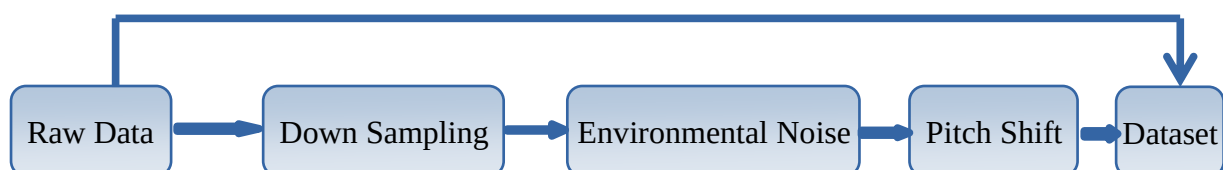


Figure 17. Data augmentation process

4.2.3 Labelling

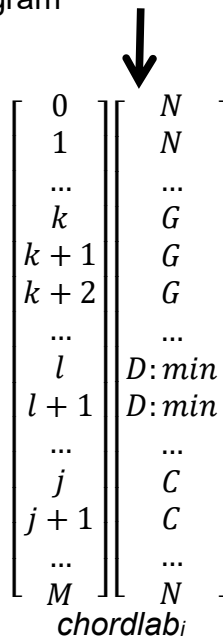
Moving forward with the annotation data, for the labelling task, each audio frame from the frequency data, has to be mapped with an appropriate chord label.

for each x in X_i :
 if $t(x) > label_start$: $x_label = C_j$
 if $t(x) > label_end \rightarrow next_label$

Annotation				Spectrogram				
	<i>Starts</i>	<i>Ends</i>	<i>Chord</i>					
0	0.0	1.2	<i>N</i>	0	0.0	0.0	...	0.0
1	1.2	2.4	<i>G</i>	1	0.0	0.0	...	0.0
2	2.4	7.1	<i>D: min</i>	2	0.2	1.5	...	0.6
3	7.1	12.9	<i>C</i>	3	0.9	4.2	...	9.2
4	12.9	14.0	<i>A: 7</i>	4	1.1	2.6	...	10.1
5	14.0	18.8	<i>G</i>	5	6.7	3.1	...	10.6
...
K	235.0	242.2	<i>N</i>	M	0.0	0.0	...	0.0

Track_i

K: number of chords in Track_i
 M: number of time steps in spectrogram



It is widely known, that in neural networks, character labels can not be used, they have to transform into one hot encodings or a similar numerical value. In order to present these chords in one hot encodings, the simplest way is to create an encoding for each unique chord that appears in the dataset. In this case, there are 407 unique chords without and 1.500 with data augmentation. So, in the process of creating one hot encodings of length 1.500, it would amount to an insanely large output space for the model. Thus, an algorithm was developed, that took each chord label and created a chord vocabulary, with which it could represent the majority of the chords inside the dataset, in a representation that could have the strongest positive influence on the model's predictions.

4.2.4 Chord Vocabulary

Later on, the chords were separated in distinct qualities. First there is the chord root, bass note (inversion), triad mode, fourth note. The dataset used is very small in order to add extra notes 9ths, 11ths and 13ths since as it will be evident later on the chapters, there is difficulty predicting the fourth note existence.

Table 5. Chord dictionary

Dimension	Meaning
0-13	Root
0-13	Bass
0-7	Triad
0-5	Fourth

Specifically,

$$Root \in \{C, C\#, D, D\#, E, F, F\#, G, G\#, A, A\#, B, N, X\}$$

$$Bass \in \{C, C\#, D, D\#, E, F, F\#, G, G\#, A, A\#, B, N, X\}$$

$$Triad \in \{Major, Minor, Diminished, Augmented, Sus2, Sus4, N, X\}$$

$$Fourth \in \{dim7, min7, maj7, maj6, N, X\}$$

Component Domain Spaces

- N = silence
- X = unknown

Depicted below is the representation of this chord vocabulary and how it is implemented in practice, as well as how all the different components are separated.

Chord	Root	Bass	Triad	Fourth Note
N	N	N	N	N
F:maj6	F	F	Major	maj6
C	C	C	Major	N
G	G	G	Major	N
A:min	A	A	Minor	N
D:min7/4	D	G	Minor	min7
F:maj9	F	G	Major	N
C/7	C	B	Major	N
C/5	C	G	Major	N
F	F	F	Major	N
Ab	Ab	Ab	Major	N
Bb	Bb	Bb	Major	N
Ab/7	Ab	G	Major	N
Ab/b7	Ab	F#	Major	N
F:7	F	F	Major	min7
B	B	B	Major	N
A	A	A	Major	N
E	E	E	Major	N
E:7	E	E	Major	min7
D	D	D	Major	N
D:7	D	D	Major	min7

Chord	Root	Bass	Triad	Fourth Note
N	12	12	0	0
F:maj6	5	5	1	4
C	0	0	1	0
G	7	7	1	0
A:min	9	9	2	0
D:min7/4	2	7	2	2
F:maj9	5	7	1	0
C/7	0	11	1	0
C/5	0	7	1	0
F	5	5	1	0
Ab	8	8	1	0
Bb	10	10	1	0
Ab/7	8	7	1	0
Ab/b7	8	6	1	0
F:7	5	5	1	2
B	11	11	1	0
A	9	9	1	0
E	4	4	1	0
E:7	4	4	1	2
D	2	2	1	0
D:7	2	2	1	2

Figure 18: Chord Vocabulary representation in practice

4.2.5 Chord analysis

It is important to provide detail analytics for the chord labeled data that exist on the datasets used. It is important to show what data are represented and used to train and test this thesis in order to be able to understand certain forthcoming such as class imbalance.

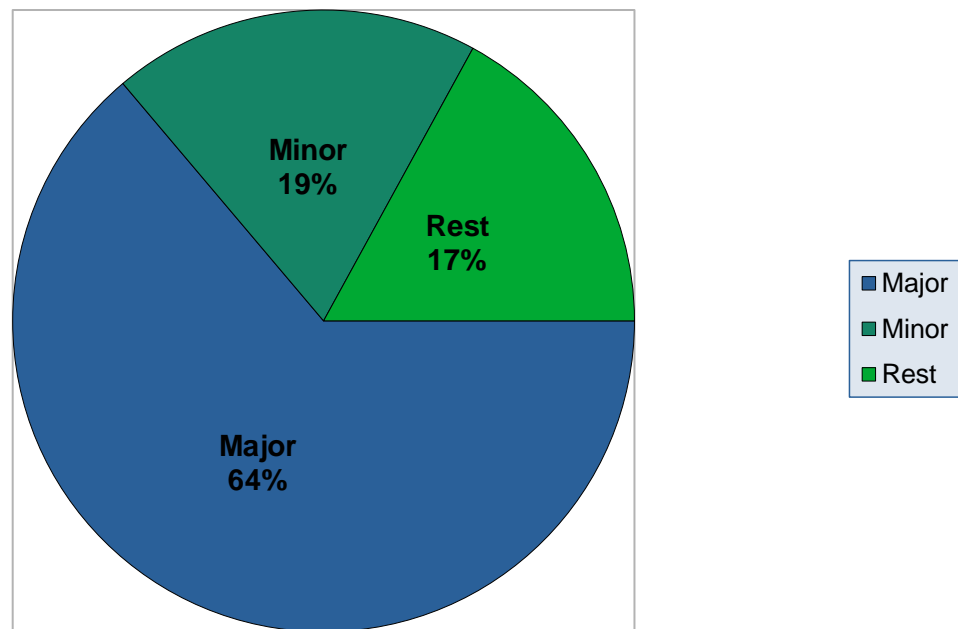


Figure 19. chord modes

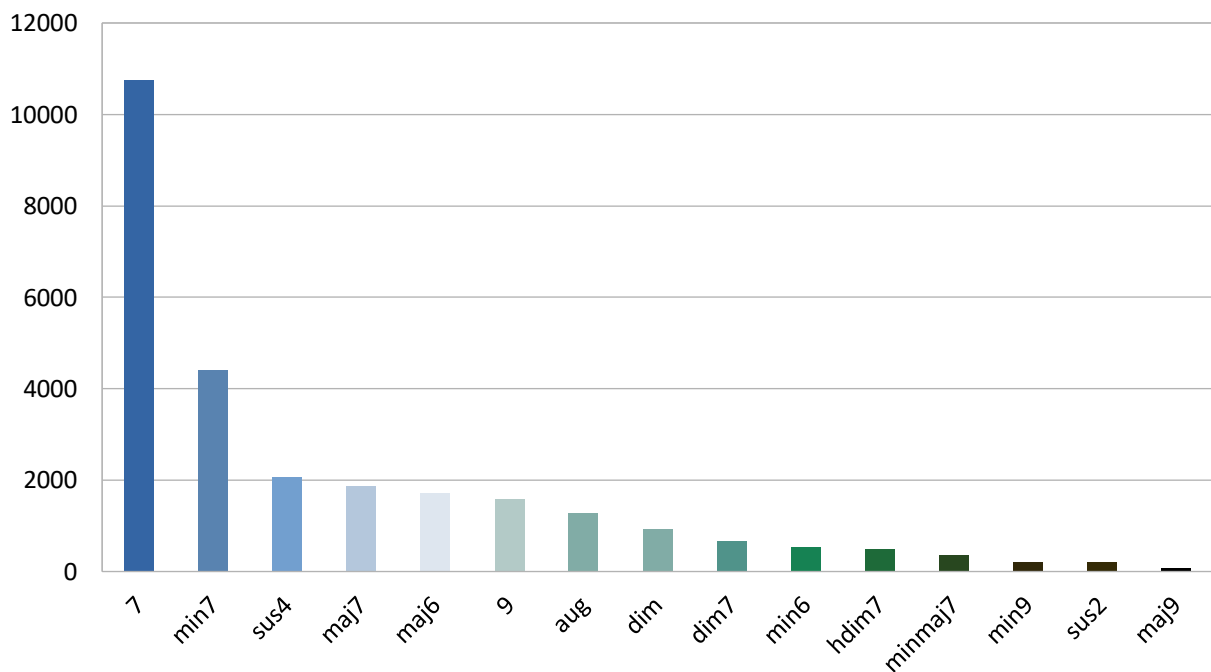


Figure 20. Non popular modes appearances

Looking at those numbers, it is evident that some classes are underrepresented, thus the problem of class imbalance rises. This problem is addressed later in this thesis.

4.2.6 Slicing

One step before training, is a technique used with which it sliced all of the training data, into small chunks of 300-time steps → 22 seconds. Because the models used are based on a bidirectional LSTM layer, it is vital to have small chunks of time series, in order for the LSTM to learn better and more efficiently.

When used the M2 architecture with slicing, there was a 5% increase in the model's accuracy.

4.3 Model – Training

In this chapter, the variety of different architectures that were used is going to be presented. Started from a very simple architecture and built the way up through a more complex one.

4.3.1 Models

M1.

Predicting with a Recurrent Neural Network Architecture (RNN). The first method used was a 2-layer recurrent neural network of Bidirectional LSTM. Even though bidirectional LSTM is a very powerful architecture for time series, it couldn't perform above a certain upper bound that it reached.

Generally, it is a very simple model that is trained very easily, and with small amount of data it reached a decent accuracy on some tasks. In this model there is only a 2-layer bidirectional LSTM with a dropout after each layer used, in order to decode the input features into feature vectors that together with a fully connected layer, will have the ability for a linear separation between root, bass, triad and fourth note.

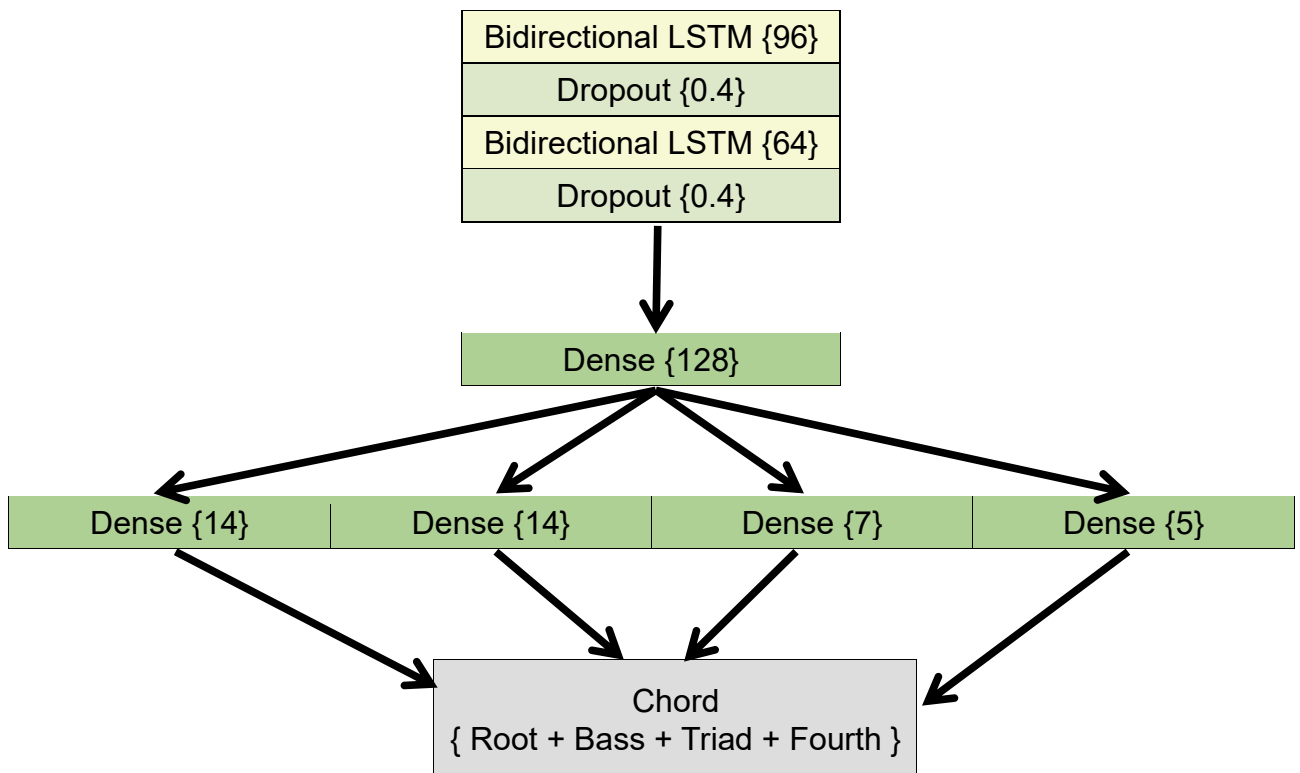


Figure 21. Recurrent Model Architecture M1

Model description

Table 6. M1 Model description and parameters

Layer	Output Shape	Parameters #
Input	[slice_size, 192]	0
Bidirectional	[slice_size, 192]	221952
Bidirectional	[slice_size, 128]	131584
Dense	[slice_size, 128]	16512
Activation (sigmoid)	[slice_size, 128]	0
Dense	[slice_size, 14]	1806
Dense	[slice_size, 14]	1806
Dense	[slice_size, 8]	1032
Dense	[slice_size, 6]	774
Activation (softmax)	[slice_size, 14]	0
Activation (softmax)	[slice_size, 14]	0
Activation (softmax)	[slice_size, 8]	0
Activation (softmax)	[slice_size, 6]	0

Total parameters: **375,466**

Trainable parameters: 375,466

Non-trainable parameters: 0

This model works, by taking as an input the 192-dimensional shaped vectors of the logarithmic frequency spectrograms and treat it exactly such as time series. Each time step in the input vector is a 192-dimensional variable, and the first layer will search for contextual information through time on them. A bidirectional RNN was used for this task, because in chords it matters what follows each value, not only what was there before. After the first layer, another Bi – LSTM layer is stacked in order to allow for greater model complexity. As the data is increasing, a simple model with small complexity will not be able to learn well enough.

Following the LSTMs is the general fully connected layer, which also benefits from the stacked LSTMs since it will not fall into certain patterns over time.

In order to achieve a multiple output model, the usage of 4 different fully connected layers gave good results, each for every output, that classifies with SoftMax activation the class that is selected from each domain.

M2.

As it was described in Chapter 2, convolutional networks work well with grid-like topology data that have a spatial relationship. Spectrograms, on the M1 model were interpreted as time series of vectors, with 192 features. In this model, spectrograms will be interpreted as images. To do that, convolutional layers are summoned before the recurrent layer, in order to provide feature extraction.

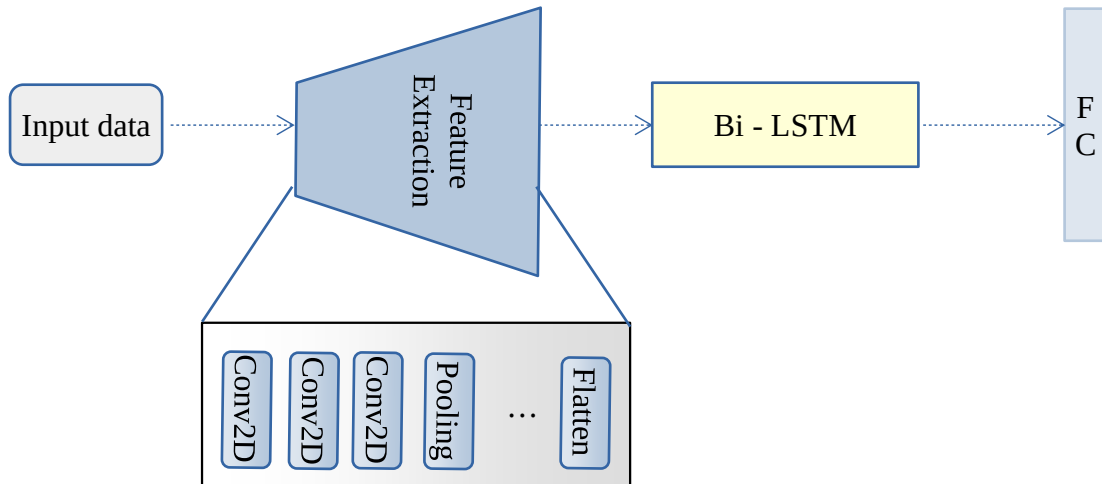


Figure 22. Model 2 Architecture

These convolutional layers will have the ability to map the spectrogram's data into an output vector, where this vector is going to be the input for the bidirectional LSTM layer which will discover all the contextual information through time.

For the convolutions there are 2 different techniques that can be applied on audio.

- 1D Convolutions
- 2D Convolutions

Both techniques provide very good results, but they both have their differences.

First of all, 1D convolutions retain only the temporal layout of the spectrogram. On the other hand, 2D convolution layers, among the temporal layout, they also retain the frequential layout, by giving attention to neighboring frequency bands. With that in mind, means that the same filters will be applied on different frequency offsets. Thus, each output vector will be a 2D feature map of the input spectrogram.

1D convolutions understand the difference between a regular image, and a spectrogram and how different in context they are. They can learn and extract features, that they can occur at any time. But, 2D convolutions, not only learn features can occur at any time, but they assume that they can occur at any frequency – pitch, shiftable across all the frequency spectrum.

With all the above in mind, the designed network, consists of 2D convolutions as a feature extraction tool, in order to extract local features. The following recurrent layer will then have a chance to decode the features spanning across the time variable.

Also, to note, 2D convolutions are a better choice than 1D convolutions for music analysis, as it is indicated by Lostanlen and Cella [16]. After thorough experimenting, the basis of the CRNN network is presented below.

The kernel has a 3×3 dimension and an incremental filter size from 16 to 64. Choosing an incremental filter size is widely used in CNNs due to the hierarchical value that they give to the context. Example, smallest features combined form a bigger one – in this case, a major third and a perfect fifth, form a major triad chord.

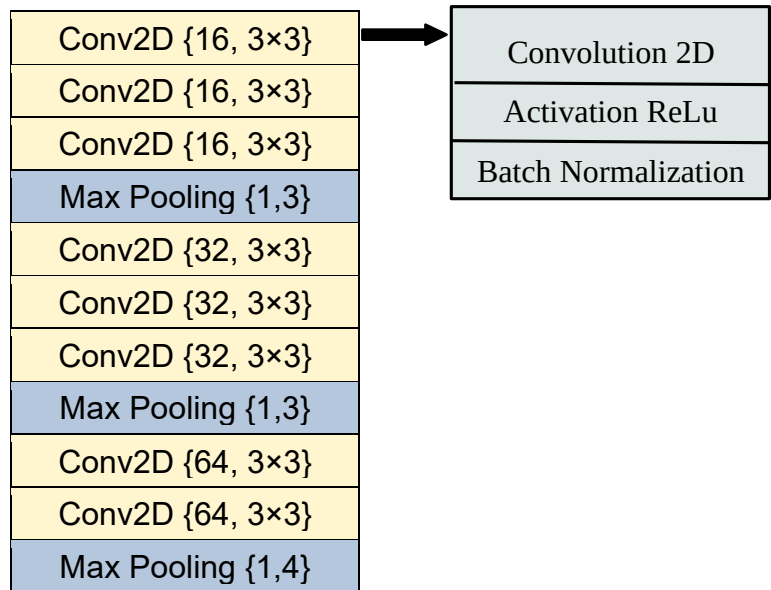


Figure 23. Part 1 – Feature Extraction FE1

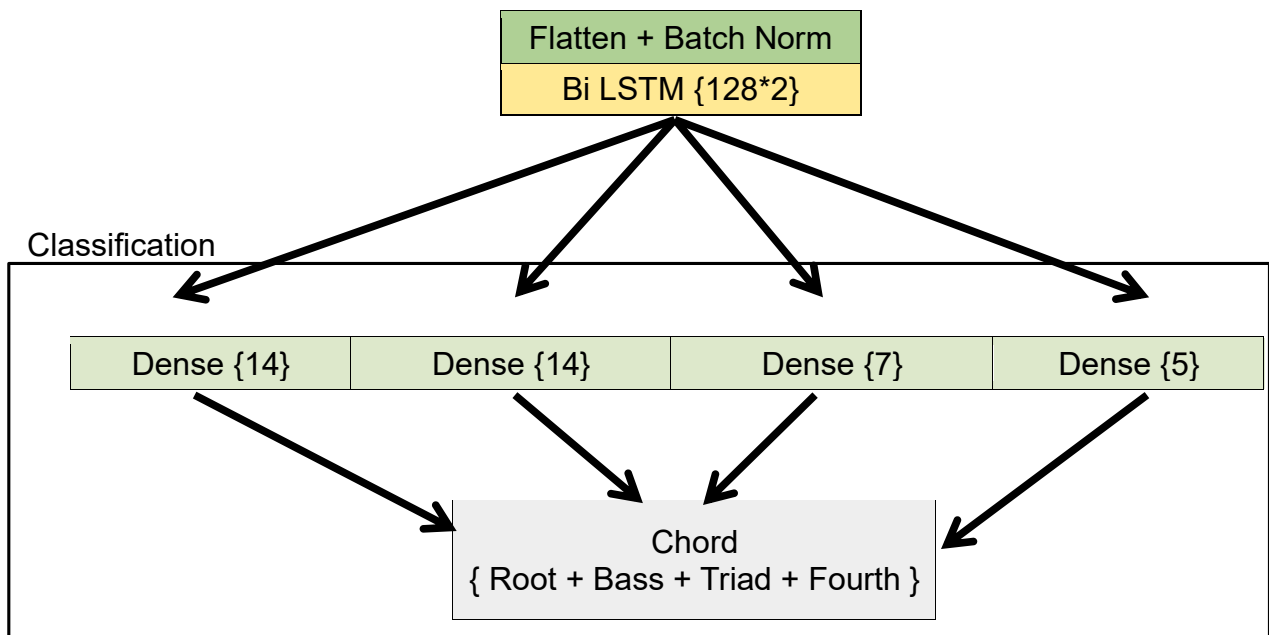


Figure 24. Part 2 – Bi-LSTM and Classification

Using the above two parts combined they achieved a 2.3 minimum loss at the 14th epoch, where the model started to overfitting, providing no better results.

In turn, a model with a higher complexity was chosen, adding another convolutional block with 128 filters.

Conv2D {16, 3×3}
Conv2D {16, 3×3}
Conv2D {16, 3×3}
Max Pooling {1,3}
Conv2D {32, 3×3}
Conv2D {32, 3×3}
Conv2D {32, 3×3}
Max Pooling {1,3}
Conv2D {64, 3×3}
Conv2D {64, 3×3}
Max Pooling {1,4}
Conv2D {128, 3×3}
Conv2D {128, 3×3}

Figure 25. Part 1 – Feature Extraction FE2

This way, the minimum loss was dropped to 2.1 and the model was scaled better, performing better at the most complex tasks – triads and fourths. The only problem with the model, is that the last convolutional layers, contain large amount of parameters, and also provide very big feature matrices. Thus, when the LSTM takes those vectors as inputs, it needs also a huge amount of parameters.

In order to solve this problem having acquainted all the extra power from the 128 filter convolutions, I had to add another layer, which would reduce the dimension from 128 to 64.

For this task, 3 different strategies were approached. First strategy was to add a Convolution Layer with filter size 64. The second one, was to try and use a fully connected layer, but not for classification, but to reduce the size of each feature matrix. The last one, was to use the pooling layers, in order to reduce the amount of feature matrices, and not the number of features in each matrix.

FE1
Conv2D {128, 3×3}
Conv2D {128, 3×3}
Conv2D {64, 3×3}

Figure 26. Part 1 – Feature Extraction FE3

FE1
Conv2D {128, 3×3}
Conv2D {128, 3×3}
Dense {64, 3×3}

Figure 27. Part 1 – Feature Extraction FE4

Those two models are the 2 best performing models. FE3 uses an extra convolutional layer with filter size equal to 64 in order to minimize the parameters that go into the Bi-LSTM.

Table 7. Parameter Comparison

FE2	FE3	FE4
1,107,722	852,810	787,018

The difference between the last layers is that a fully connected layer, has only

$$128 \times 64 + 64 = \mathbf{8,256}$$

parameters
(+64 is for the bias)

because it is the same used for all the features matrices produced by the convolution.

On the other hand, the convolutional layer has

$$(3 \times 3 \times 128 + 1) \times 64 = \mathbf{73,792}$$

parameters.

$(n \times m \times l + 1) \times k$, where n, m = kernel size, l is the input features, +1 for the bias and k is the output features.

Between the FE3 and FE4 models, as it is evident from the below graphs, they perform almost the same.

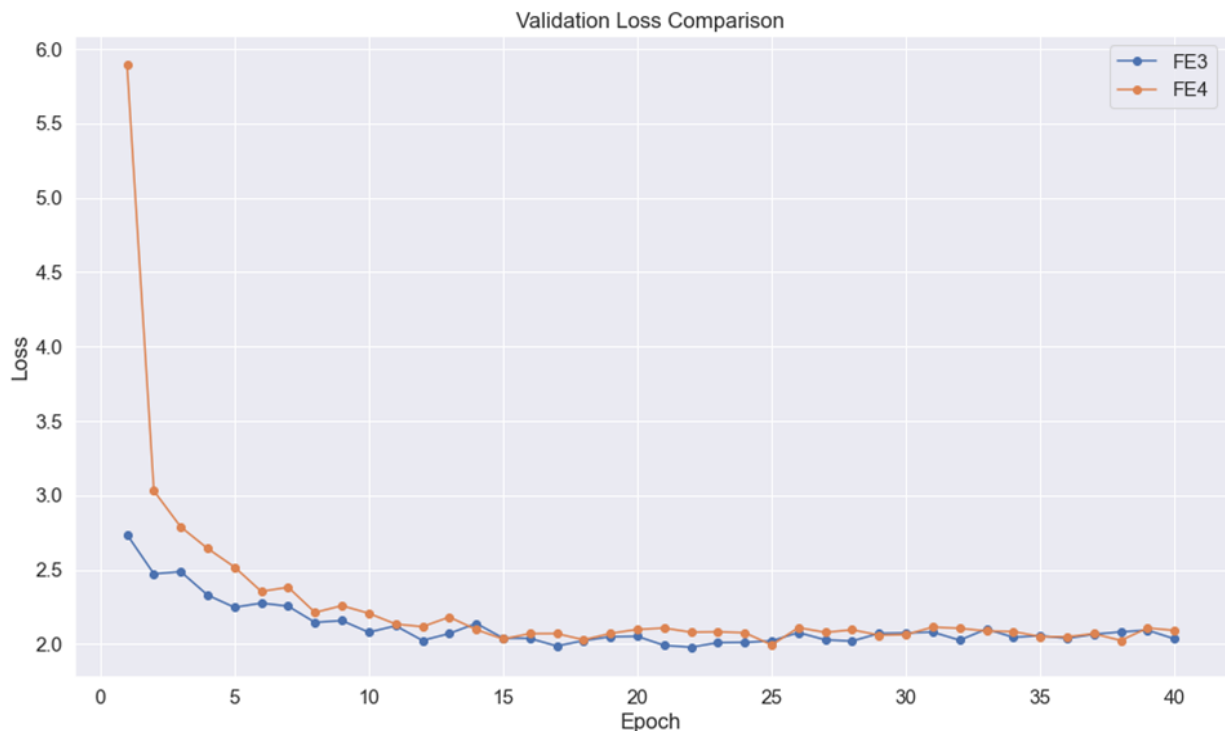


Figure 28. Validation Loss Comparison FE3 and FE4

On the other hand, the strategy with using the pooling layers differently, leading to fewer feature matrices, did not work as expected as it started overfitting after 20 epochs, hitting a 2.2 minimum loss. Although the training was faster, and the model had fewer parameters equal to 581,386 the models FE3 and FE4 seemed to work better with this validation set.

Lastly, a strategy that is commonly used, is using fully connected layers to decode the feature extraction into simpler feature vectors, generalizing the information extracted by the CNN.

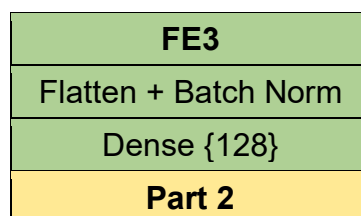


Figure 29. Fully Connected Vectorization FCV1

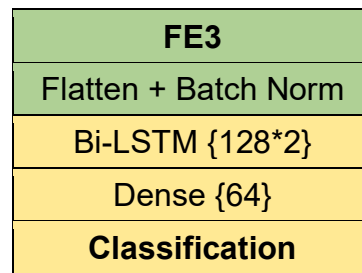


Figure 30. Fully Connected Vectorization FCV2

Those 2, are the last models that came to trial, and their performances will be presented in detail on the next chapter.

Generally, in all the models, each convolutional block consists of the convolution followed by a ReLu activation and a batch normalization. There are certain applications, mostly in deep neural networks, where batch normalization is one of the most important foundations of the network. Using batch normalization makes our model capable to learn at higher learning rates without facing problems with the initialization of the weights. [17]

4.3.2 Training

The models were trained on a 2x Tesla K80 GPU and 32GB RAM on the Google Cloud Platform.

- Batch Size = 32 (when a sample is a 22 second part of a track). The number of training examples in one forward/backward pass.
- Optimizer = Adam
 - Learning rate = 0.0001
 - Beta1 = 0.900
 - Beta2 = 0.999
 - Epsilon = 10^{-8}
- Loss = the sum of the 4 individual losses of the model [root, bass, triad, fourth] each computed with categorical cross entropy.

$$Loss_{crossentropy} = - \sum_i y_i \log(\hat{y}_i)$$

$$Loss = \sum_1^4 Loss_{CrossEntropy_i}$$

- Epochs
 - CRNN: 40
 - RNN: 80

4.3.2.1 Model M1 Experiments

Callback history of loss and accuracy for both train and validation sets.

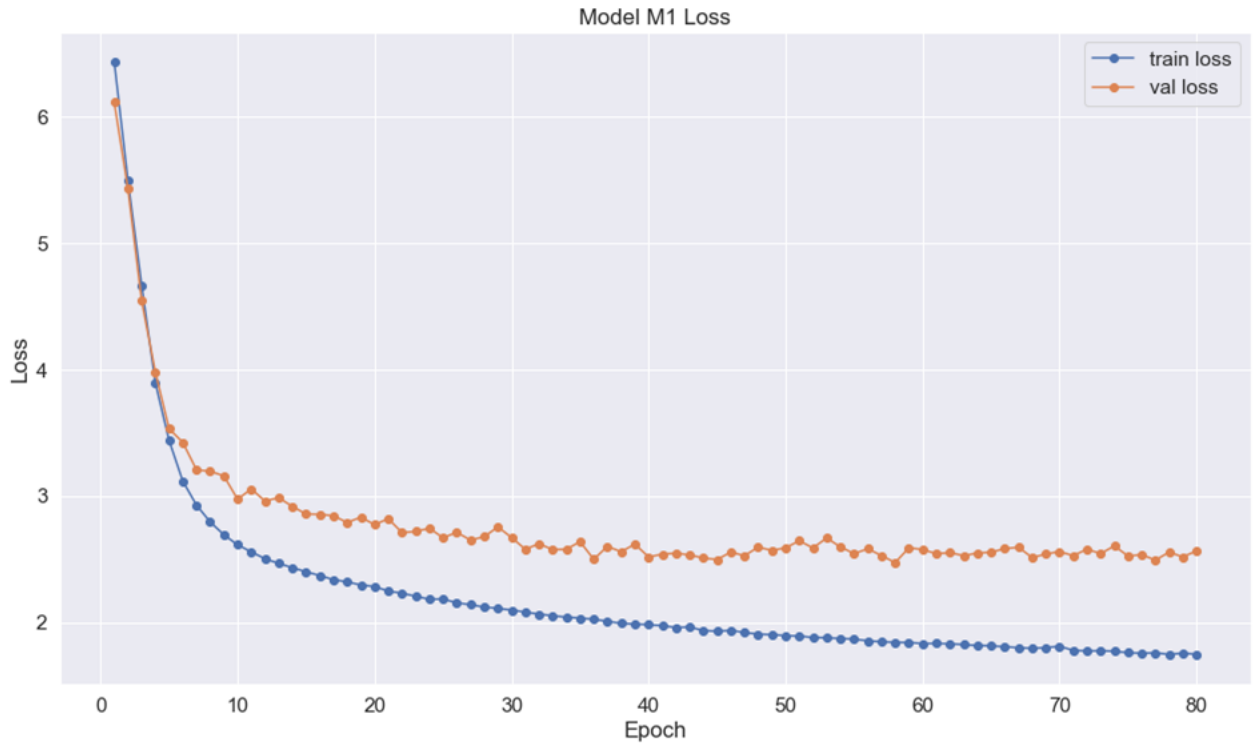


Figure 31. Callback History Loss M1

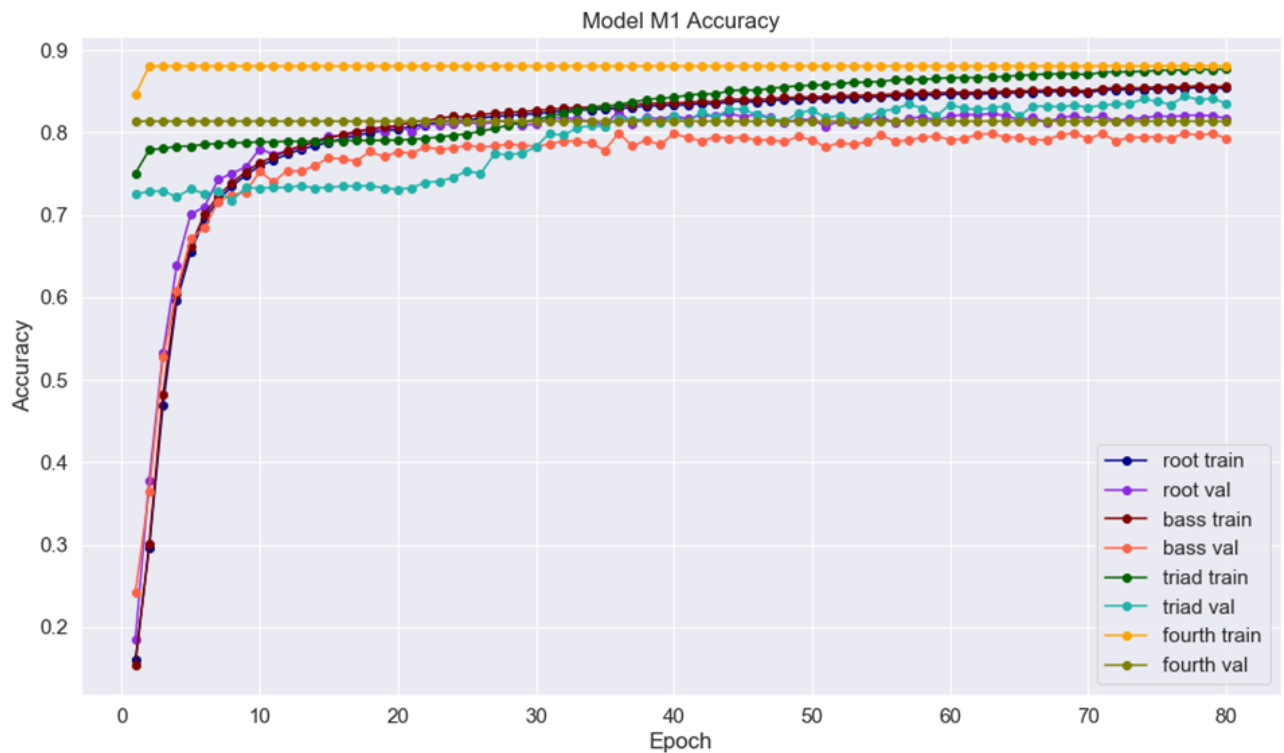


Figure 32. Callback History Accuracy M1

As it is clearly depicted from the figures above, this recurrent model was able to scale and identify root and bass notes all the way up to an accuracy of 80%. To provide more detail about those 2 classification tasks, below is provided the confusion matrix for the root prediction task.

Root Task

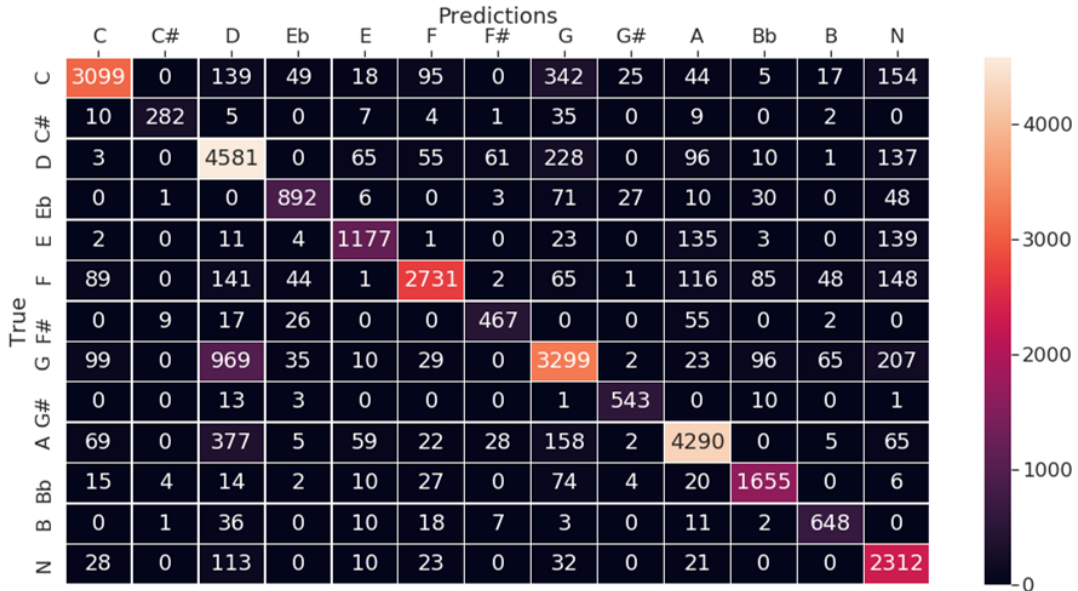
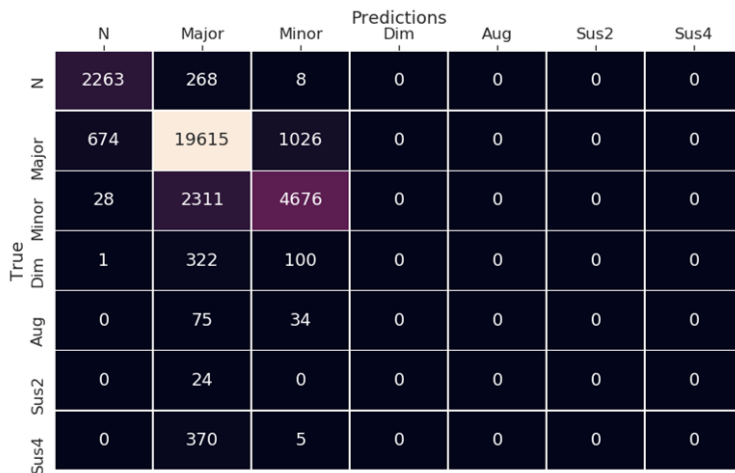


Figure 33. Root Task Confusion Matrix M1

On the other hand, triads and fourth notes are stable on an accuracy percentage over 80%. Accuracy is not a representative metric for this task. Due to the imbalance of classes, and the non – existence of a fourth note most of the times, as well as most of the chords being major, other techniques must be used, in order to provide a good understanding of how this model behaved. For this reason, I will present the confusion matrix, a table that shows the performance of a classification task. It is a way to visualize the performance of the model. Most times confusion matrices are seen on binary classification tasks. Below, a multi class confusion matrix is used, that visualizes the True class vs the Predicted class.

Triad Task



As it is evident, mostly all silences were identified correctly. The distribution of chords in this matrix, tends to predict major.

- Major 94.7%
- Minor 60.3%
- Diminished 0%
- Augmented 0%
- Sus2 0%
- Sus4 0%

Figure 34. Triad Task Confusion Matrix M1

Fourth Task

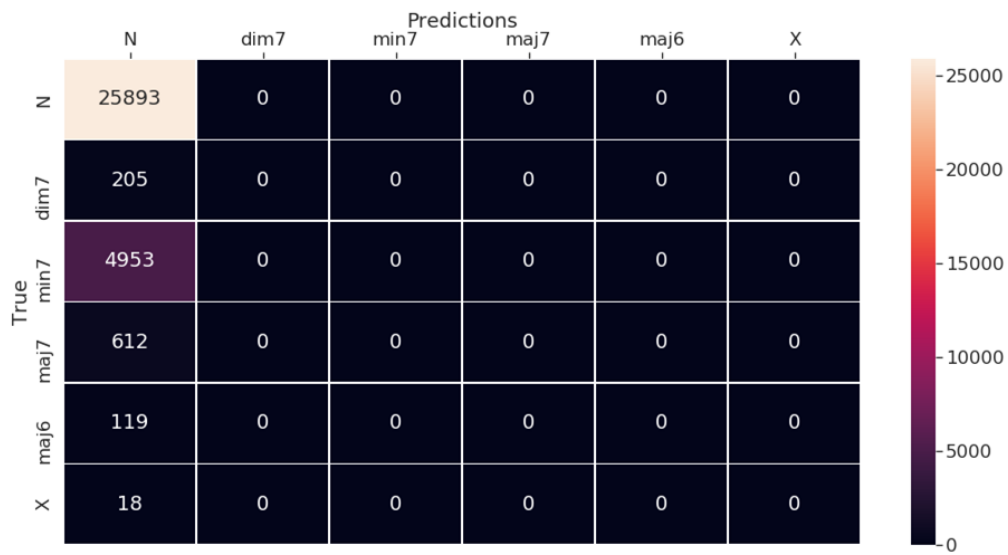


Figure 35. Fourth Task Confusion Matrix M1

On the fourth note prediction task, it is evident that the model has completely failed to identify a single one fourth note, besides silence. Although the train prediction is 94% and the validation prediction is 81.42% it classified as silence, all dim7, min7, maj7 and maj6 chords.

Finally, after thorough experimenting, concluded that a better result could not be achieved only by working with recurrent layers. There was a need to implement an architecture that would have the ability before the recurrent layers, to extract features from the input. Thus, a recurrent convolutional architecture neural network was implemented, M2.

4.3.2.2 Model M2 Experiments

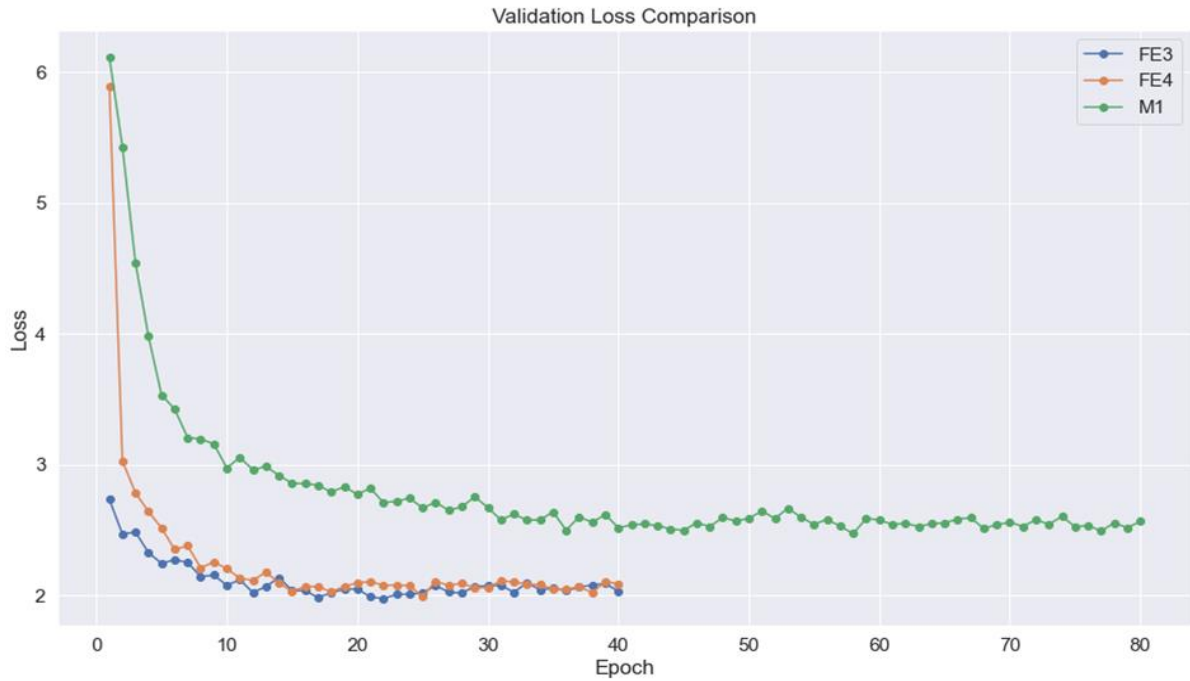


Figure 36. Comparison CRNN with RNN

The figure, clearly depicts the difference that made the feature extraction part of the network, dropping the validation loss significantly.

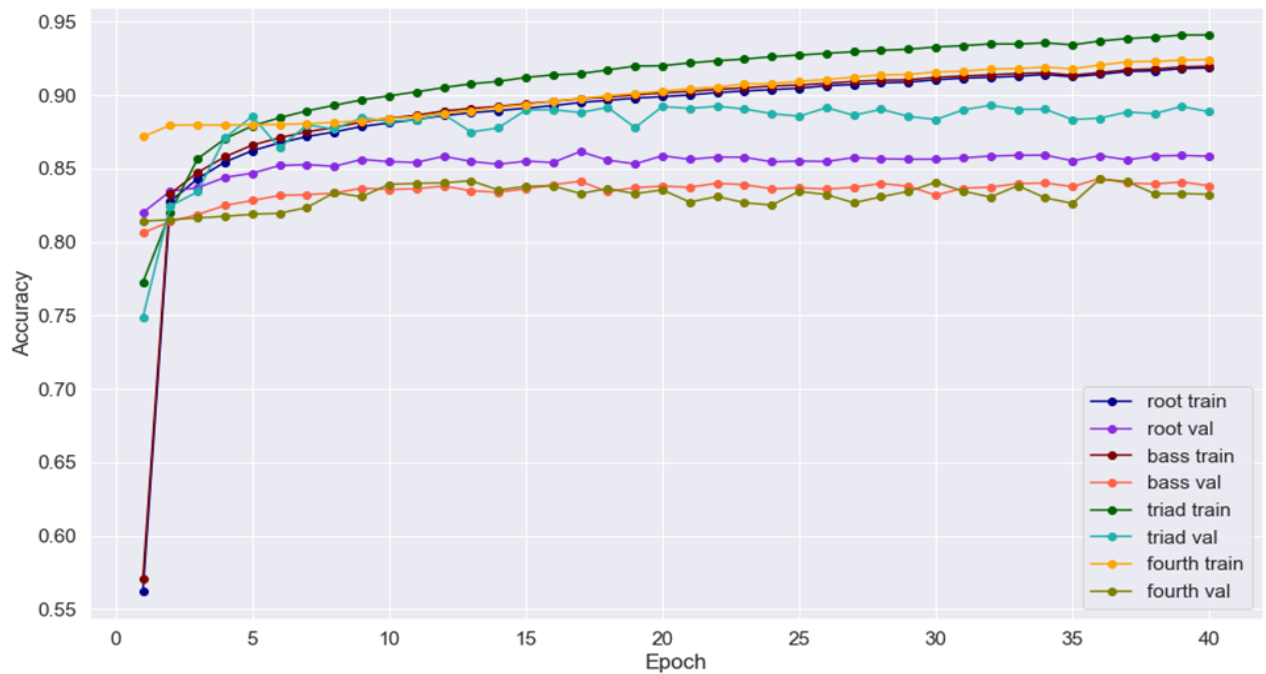


Figure 37. Callback History Accuracy FCE3

Triad Task

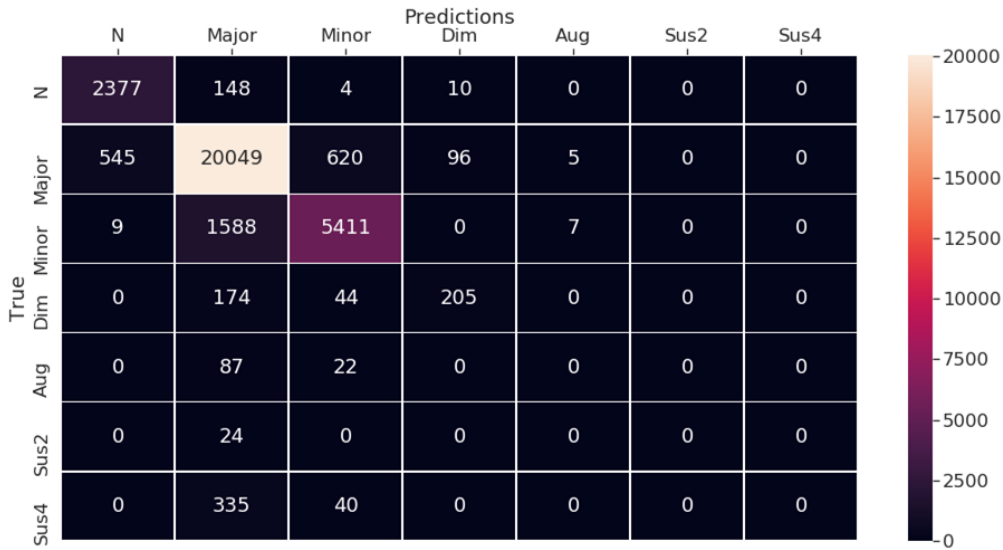


Figure 38. FCE1 Triad Task Confusion Matrix

On the other hand, with the addition of the 128 filter size convolutions the accuracy increased as it is depicted in the matrices.

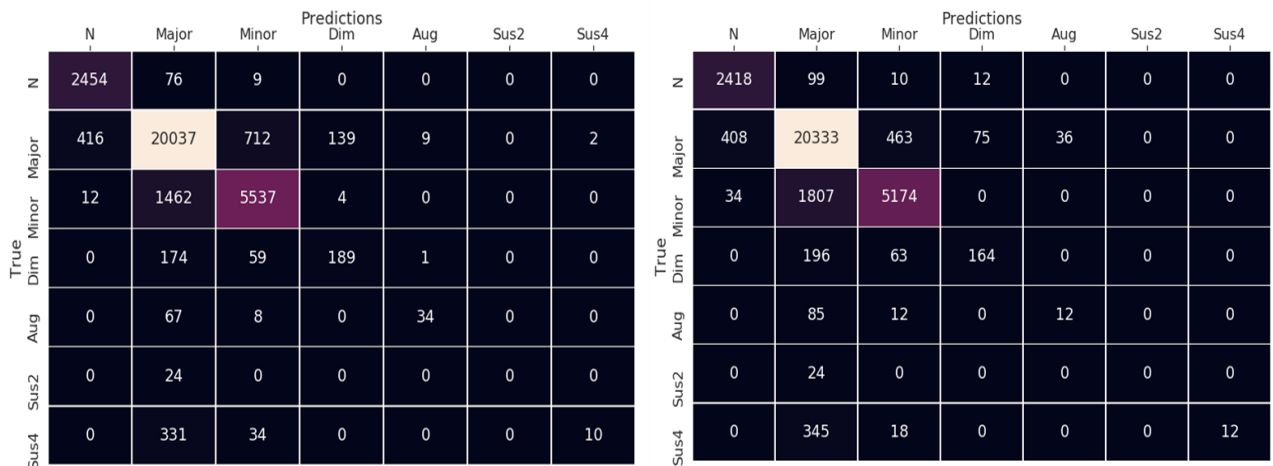


Figure 39. Left FCE3, Right FCE4 confusion matrices triad task

FCE4 and FCE3 have a little variation on the minor and major chords correct predictions and it is understandable. The real difference comes to show when the latter two models, could predict some of the least represented chords in the dataset – augmented and sus4.

Table 8. Triad Task model comparison accuracy

	FCE1	FCE3	FCE4
Major	94.06%	94.00%	95.39%
Minor	77.13%	78.93%	73.75%
Diminished	48.46%	44.68%	38.77%
Augmented	0.00%	31.19%	11.00%
Sus2	0.00%	0.00%	0.00%
Sus4	0.00%	0.03%	0.03%

Fourth Task

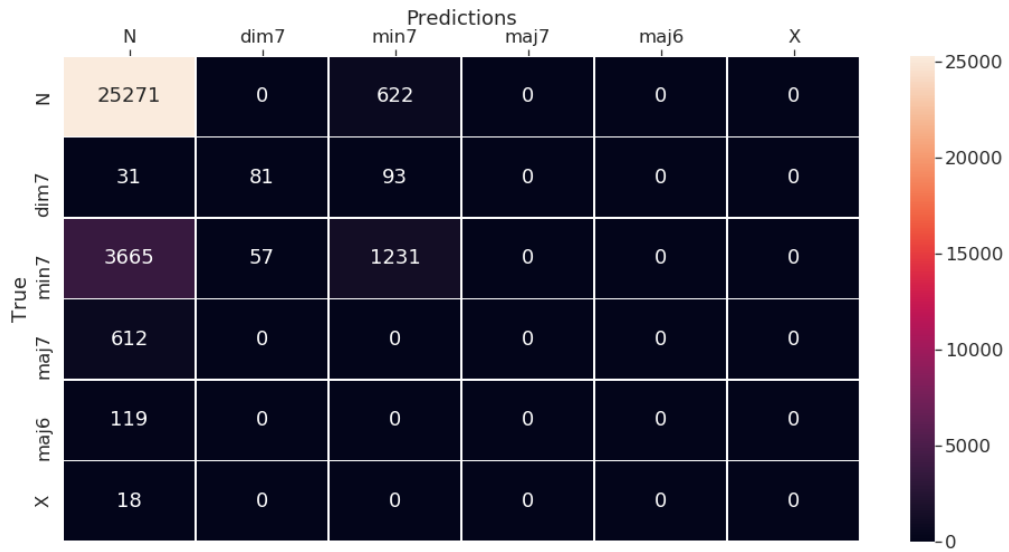


Figure 40. FCE1 Confusion matrix on fourth task

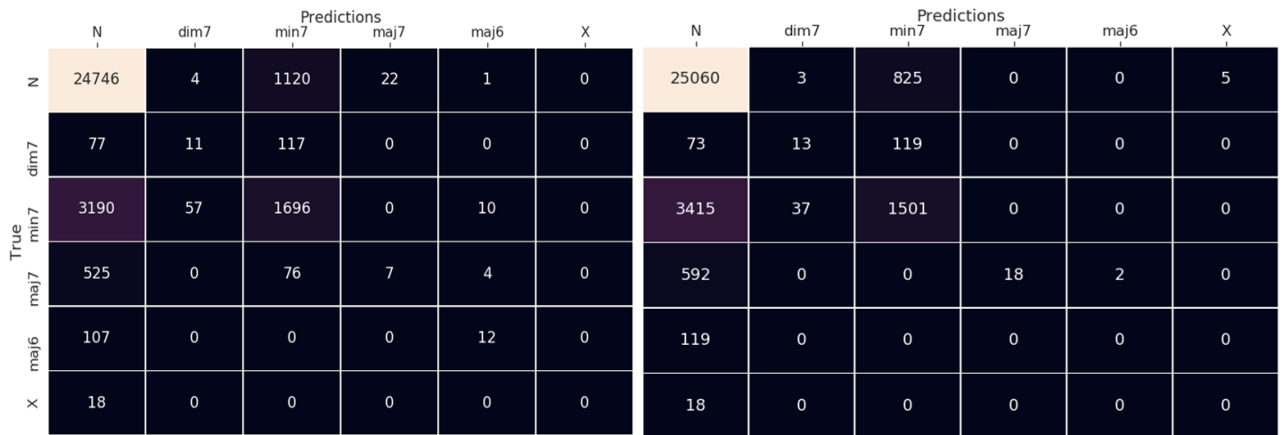


Figure 41. Left FCE3, right FCE4 Confusion matrix on fourth task

Table 9. Fourth accuracy model comparison

	FCE1	FCE3	FCE4
N	97.59%	95.57%	96.78%
dim7	39.51%	0.05%	0.06%
min7	24.85%	34.24%	30.30%
maj7	0.00%	0.02%	0.04%
maj6	0.00%	0.10%	0.00%

4.3.3 Imbalanced classification

As it is evident from the data analysis done on chords, and confirmed by the training process, most chord classes are underrepresented, a problem called imbalanced classification. When the models see 10K major chords but only 1000 minor chords it will not be able to classify the minor chord as minor and will label them as major.

To illustrate this further, with a dataset of 10K major chord and 1000 minor, and only those 2 classes, when a track containing 10 major chords is submitted for evaluation, the model will have 100% accuracy, but when a track with 8 minor and 2 major chords is submitted for evaluation the accuracy will be roughly 20%.

In order to solve this problem, there are lot of different solutions and metrics, all relevant to the dataset and problem on hand. In the case of this thesis, 2 different things can be applied.

1. Create more instances of the underrepresented classes or delete some instances of the overrepresented one.
2. Apply weighted loss training. This is a process, where the loss is not influenced equally by each class that is not classified correctly. In the simple case of not weighted loss training, imagine that each class has a weight equal to 1, and the loss is influenced by this weight \times classification error. It is feasible to apply a different weight on each class, and thus a misclassified minor class will influence the cross-entropy loss more than a misclassified major.

The mindset it to punish the model more for wrongly classifying a chord that is not seen so many times, and not apply much influence when there a wrong classification of a class that exist several instances.

From the two previously methods, only the weighted loss training made it to implementation. In order to do that, all the input data were scanned to count instances of classes appearances. It is important to note that weighted loss training was performed only for triads and fourths. Due to data augmentation, the classes of root and bass notes were uniformly distributed.

Thus, the weight of each class is computed by,

$$\text{Weight}_{class} = \frac{\max \{ \sum_{i=0}^{\text{Alldataset}} \text{instances}_{class} \forall \text{ class} \}}{\sum_{i=0}^{\text{Alldataset}} (\text{instance}_{i==class})}$$

For the weighted loss training 2 different algorithms were used.

1. Used the below formula

$$w = \log_2(\text{Weight}_{\text{class}} + 1)$$

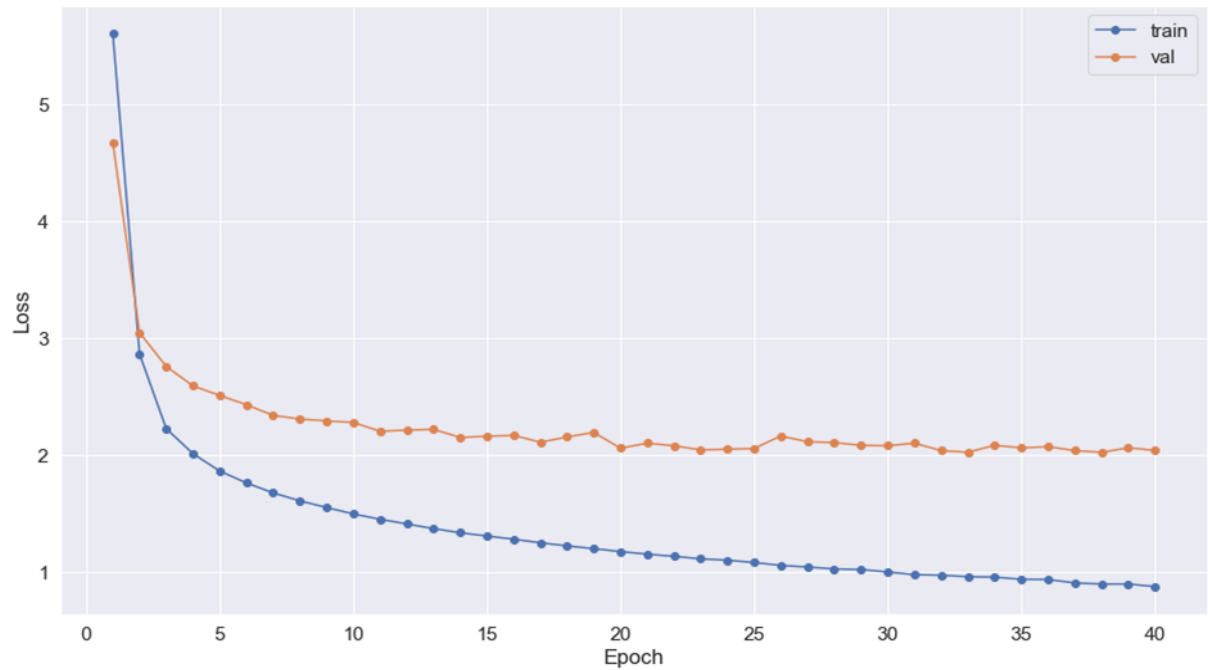


Figure 42. Loss model WL1

2. Use for each sample the corresponding class weight

$$w = \min(\text{Weight}_{\text{class}}, a) \text{ where } a \text{ is a hyperparameter}$$

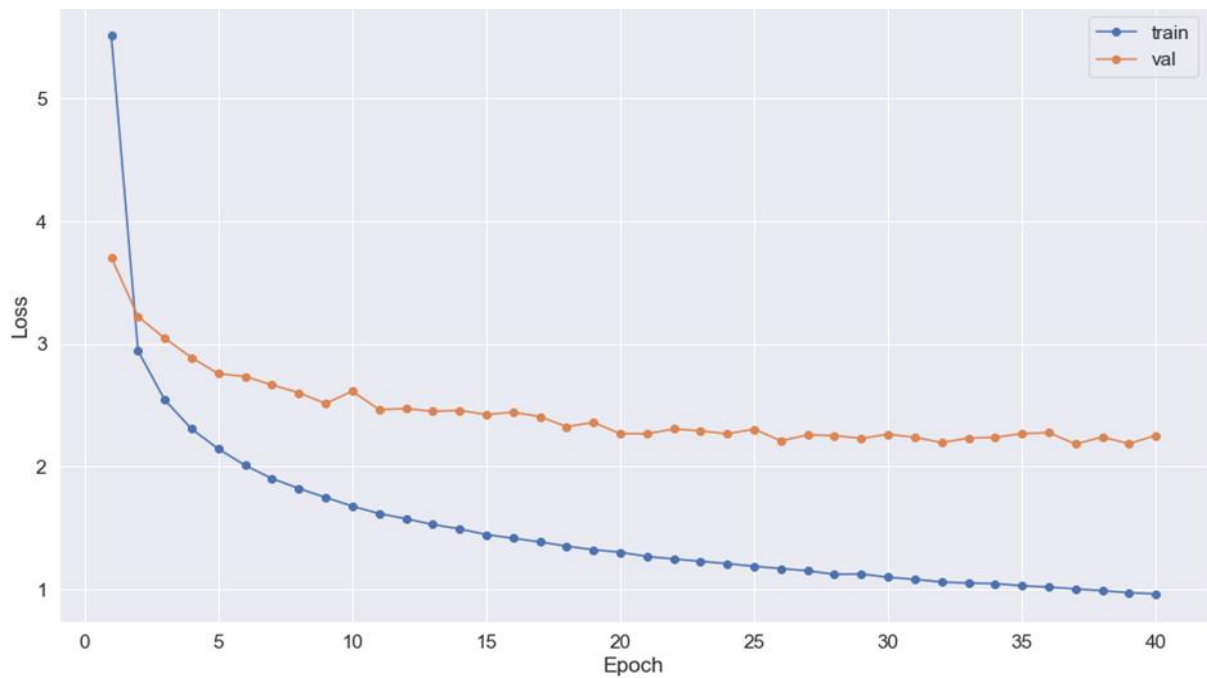


Figure 43. Loss model WL2

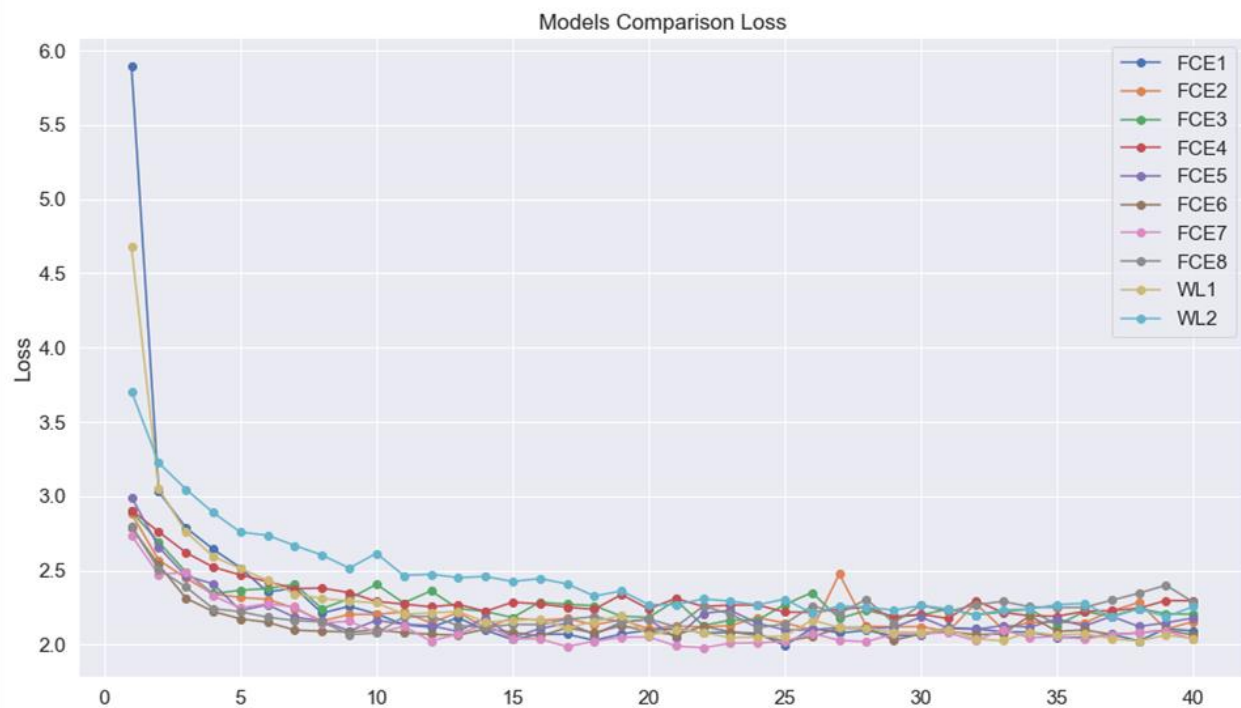


Figure 44. All model comparison

Fourth Task

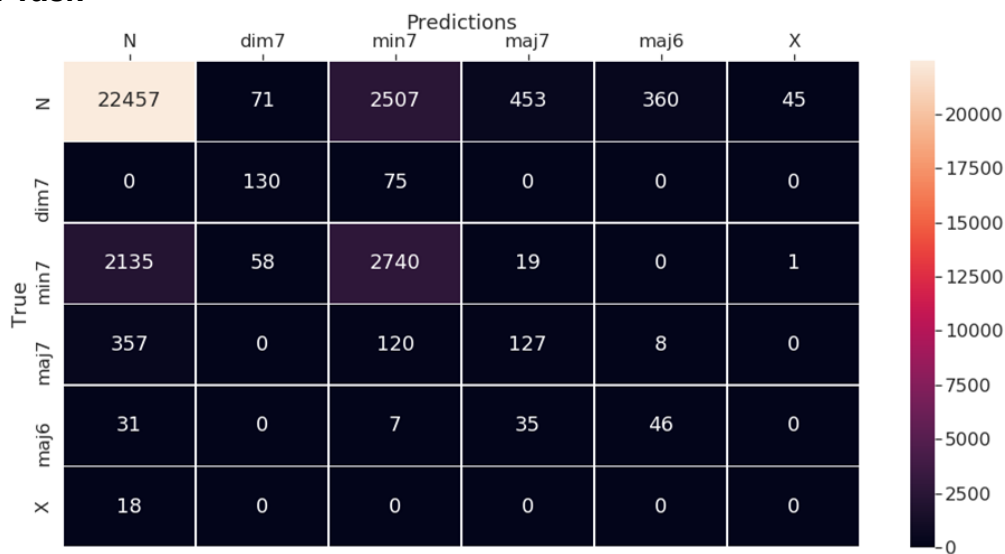


Figure 45. WL1 fourth task confusion matrix

In comparison with the other models, this is the best performing one on this task.

Table 10. Fourth accuracy weighted model comparison

	FCE1	FCE3	FCE4	WL1
N	97.59%	95.57%	96.78%	86.7%
dim7	39.51%	0.05%	0.06%	63.4%
min7	24.85%	34.24%	30.30%	55.3%
maj7	0.00%	0.02%	0.04%	20.8%
maj6	0.00%	0.10%	0.00%	38.7%

4.4 Post – Processing

Processing applied on data after the predictions, in order to smooth the results and provide a better and accurate result.

4.4.1 Heuristic Algorithm

In the models presented, the last layer before classification is the Bilinear LSTM which correlates each time step with a certain vector, and this vector is what the fully connected layer sees in order to classify root, bass, triad and fourth. These predictions due to the large amount of time steps and high sampling rate, certain noise might appear – random predictions.

In order to smooth results the necessity for a certain algorithm based on the concept of median filtering had to be implemented. The philosophy of the algorithm is to create certain entities through time and through those entities filter the random predictions with appearances ≤ 4 . Firstly, in order to create a chord entity, the root predictions are scanned, replacing the random predictions of the rest of the components.

Root Vector	Name
[0.05, 0.92, 0.03]	C
[0.05, 0.92, 0.03]	C
[0.05, 0.90, 0.05]	C
[0.12, 0.87, 0.01]	C
[0.18, 0.81, 0.01]	C
[0.66, 0.32, 0.02]	G
[0.55, 0.40, 0.05]	G
[0.12, 0.78, 0.10]	C
[0.12, 0.78, 0.10]	C
[0.05, 0.92, 0.03]	C
[0.02, 0.91, 0.07]	C
[0.02, 0.88, 0.10]	C



Applying my post processing filter is going to replace those G chords with the chord that was playing before the noise – C

To explain this further, first consider the maximum size of the “noise” to be reduced. Then consider the filter_size, and a window of size

$$\text{Window_Size} = 2 * \text{filter_size} + 2$$

It is necessary $1 * \text{filter_size}$ for the potential minor class, $1 * \text{filter_class} + 1$ for the over weighting other class, and $+ 1$ to be sure, that the filter size is not at the edges of the window. With that window, it is feasible to scan all the prediction data, and should it find that a class is over weighting some minor class that fits in the filter size, the minor class is going to be replaced with the weighted class.

4.5 Evaluation

In this chapter there is an extend analysis on the evaluation techniques used and explain what each score means, as well as presenting the complete evaluation graphs for each method used.

4.5.1 Metrics

For this thesis, were used the metrics proposed by J. Pauwels and G. Peeters [13] for the MIREX competitions for audio chord estimation. They proposed evaluation methods in a sound and unambiguous way. Those metrics involve scores for different elements of this thesis, evaluating the model accuracy for predicting:

- root of the chord
- bass note
- major / minor
- triad chord
- seventh note
- Mirex score – at least 3 notes correct

Those different metrics, all has foundation on segment evaluation, they just use different components of the reference, and evaluation to do that.



Although, if in the reference chords instead of a B:min7 existed a B:min7 the Mirex evaluation would be as below:

Reference



Estimation



Mirex Evaluation



Because Mirex metrics evaluates if the 2 chord have at least 3 same notes. B:min7 and D:min are very similar chords, and both contain {D, F, A}. B:min7 is a D:min with a B on the bass. In turn, Mirex metric sees them as similar chords.

The scores are calculated as a percentage:

$$\frac{\text{Total_of_correctly_predicted_time_frames}}{\text{Total_time_frames_of_song}} * 100\%$$

The test set is the CD1 containing 17 singles (out of 180).

Table 11. Mirex statistics all models

	Root	Third	MajMin	MIREX	Seventh	Thirds Inversion	MajMin Inversion	CSR
M1	80.44%	75.36%	69.24%	74.77%	51.33%	70.19%	65.59%	55.64%
FCE1	84.02%	80.99%	74.78%	80.75%	57.45%	75.32%	70.63%	61.24%
FCE3	84.96%	82.12%	75.88%	82.33%	58.89%	76.66%	71.91%	62.76%
FCE4	85.72%	82.51%	75.64%	82.61%	58.52%	76.36%	71.65%	62.49%
FCV1	84.11%	81.56%	75.32%	81.98%	58.37%	76.06%	71.33%	62.12%
FCV2	84.32%	81.15%	74.80%	81.42%	58.35%	75.94%	71.25%	62.05%
WL1	84.97%	81.38%	75.08%	82.99%	56.36%	76.08%	71.39%	60.08%

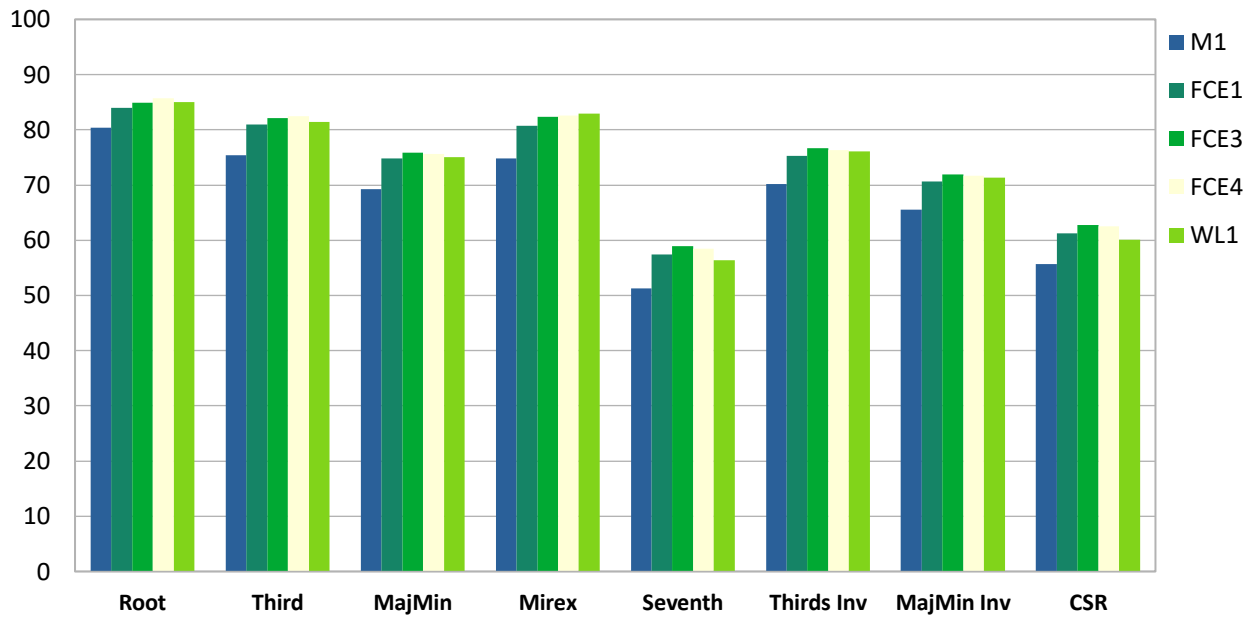


Figure 46. Mirex statistics model comparison bar chart

Although the dataset used is not sufficient to make definite conclusions, big changes in the architecture have an impact in performance. The simple recurrent model has lower Mirex statistics on all areas. Upon using the M2 architecture, with the FCE1 model that goes up to 64 filter size convolutions, there is a significant increase in the Mirex accuracy of 5 to 6 % depending on the category. On top of that, using further convolutional layers with 128 filter size, further increased the Mirex statistics by 0.5 to 1.5 %.

5. COMPARISON WITH MIREX PAPERS

The comparison between this thesis models and the models of the MIREX competition is not representative, and that is due to the fact, that the models on MIREX were trained on many more data than the presented ones, thus they are generalized over a number of different songs. As described earlier, due to the nature of this thesis and the copyrights of the songs, it is difficult to acquire the large amount of data that would be a lot helpful.

As an example, the McGill Billboard has been made public, but only several features. More specifically, all the songs with the annotations are published but with chromagrams of 36 frequency bands. For the presented models, a very different representation was used, and thus couldn't not use what the McGill university provides publicly. Quoting from the MIREX page,

“The training and testing divisions differ for the two data sets. The Isophonics has been available publicly for so long that it no longer makes sense to offer a separate training phase; as such, the entire data set will be used for testing, as in previous years. “

In turn, the statistics provided by Mirex are extracted from testing on all of the Beatles dataset. On the other hand, my scores were extracted by testing on CD1 containing all the singles of Beatles. Those songs, contain a decent representation of chords in the same analogies they appear on the whole dataset. As a result, the goal was to provide some scores that will have a minor relevance to provide a head to head comparison between state of the arts models and mine.

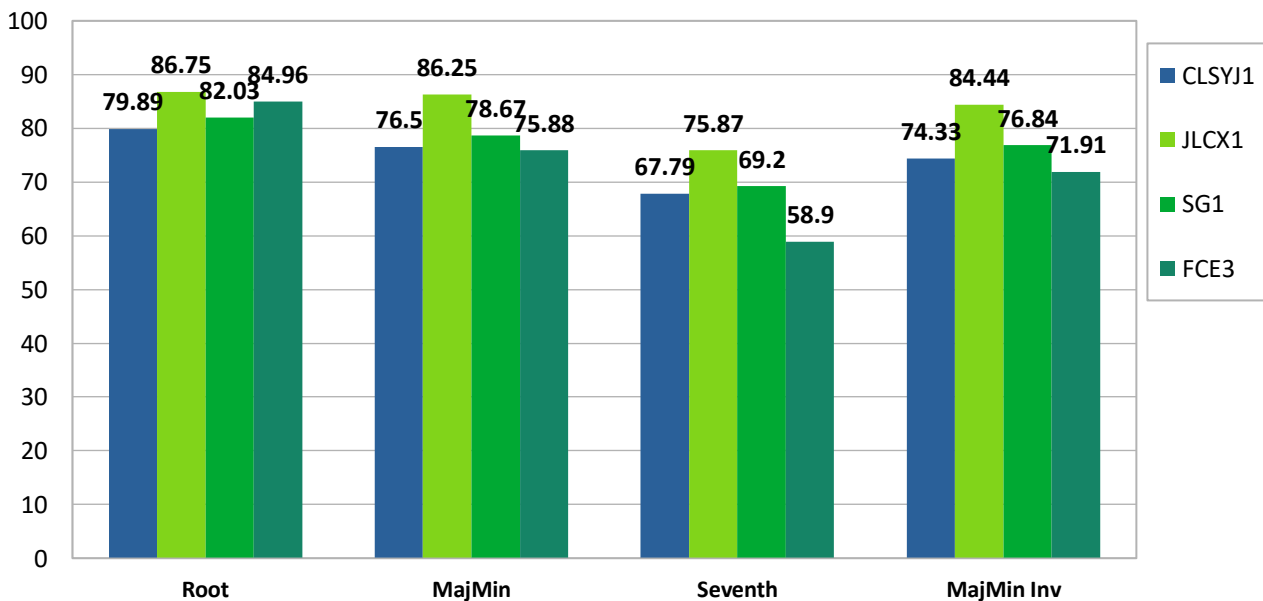


Figure 47. Mirex statistics state of the art model comparison bar chart

As it is clear from the graph, the area that my model performs most poorly against the others is the Mirex Seventh, due to Seventh being the task with the least training values.

6. FUTURE WORK

For the future, this thesis has a lot more ground for innovation and experimentation.

Firstly, having more data in hand would be extremely helpful – which due to copyright of audio tracks, was difficult to do so – in datasets like Billboard 2012, USpop2002 and Robbie Williams. With these datasets, possibly there would be a greater and more general model, that could identify chords, on a number of different styles (in the modern western pop music).

Thus, the difference between similar models like FCE3 and FCE4 might be more evident and provide with results that will lead into a position to make more certain conclusions about.

Secondly, there is ground for a more state of the art system of post processing. From Viterbi algorithm, to try and embed models like word embeddings into chord sequences, in order to filter the predictions with the most possible next chord in a sequence.

Lastly, there is the matter of the beat tracking – which is also a separate MIREX classification task – which can be used, as a secondary input, for the model to decode better the time where the chord will change.

7. CONCLUSION

To come to a conclusion, M2 architecture over performed the simple M1 architecture, and it is evident from all performance graphs and statistics. The CNN feature extraction part of the model provided the Bi – LSTM with very good feature matrices and led to decent representation of the input data for all 4 tasks. In more depth, the performance of all the models of the M2 architecture, are performing very well in the same area. It is not safe to make any definite conclusions from these results, since the training and testing datasets are limited. Nevertheless, they provide a good idea of which one could scale better on a larger dataset, with more chords of the under – represented classes. Also, the advantages of domain reduction by separating the chord recognition task in 4 different tasks are substantial.

ABBREVIATIONS – ACRONYMS

CNN	Convolutional neural network
RNN	Recurrent neural network
LSTM	Long Short-Term Memory
CRNN	Convolutional Recurrent Network
Bi – LSTM	Bidirectional LSTM
F C	Fully Connected Layer
MLP	Multi – Layer Perceptron
DSP	Digital Signal Processing
STFT	Short Time Fourier Transform
MIREX	Music Information Retrieval Evaluation eXchange
MajMin	Major Minor
CSR	Chord Symbol Recall

REFERENCES

- [1] Ian Goodfellow and Yoshua Bengio and Aaron Courville “Deep Learning” MIT Press. (2016).
- [2] Jan Schlüter “Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals” Department of Computational Perception, Johannes Kepler University Linz. (2017)
- [3] Harte, Christopher. Towards automatic extraction of harmony information from music signals. (2010).
- [4] Christon-Ragavan Nadar , Jakob Abeßer , Sascha Grollmisch “Towards CNN-based Acoustic Modeling of Seventh Chords for Automatic Chord Recognition.” SMC (2019).
- [5] Bruno di Giorgi, Massimiliano Zanoni, Augusto Sarti, Stefano Tubaro, Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony, in Proceedings of the 8th international workshop on multidimensional (nD) systems – nDS13 – November 9 – September 11, 2013, Erlangen, Germany
- [6] Vijayaditya Peddinti, Daniel Povey, Sanjeev Khudanpur. “A time delay neural network architecture for efficient modeling of long temporal contexts.” INTERSPEECH (2015).
- [7] Sak, Hasim, Andrew W. Senior and Françoise Beaufays. “Long short-term memory recurrent neural network architectures for large scale acoustic modeling.” INTERSPEECH (2014).
- [8] Felix A. Gers, Jurgen Schmidhuber, Fred Cummins “Learning to Forget: Continual Prediction with LSTM” Neural Computation (2000).
- [9] Hochreiter S. and Schmidhuber J. “Long short-term memory.” Neural Computation (1997).
- [10] Mike Schuster and Kuldip K. Paliwal “Bidirectional Recurrent Neural Networks.” IEEE (1997).
- [11] Jyh-Shing Roger Jang, Tzu-Chun Yeh, Song-Rong Lee, I Chien, Chord Estimation, Mirex 2019 submission
- [12] Junyan Jiang, Ke Chen, Wei Li, Gus Xia “Large-vocabulary Chord Transcription Via Chord Structure Decomposition” ISMIR 2019
- [13] J. Pauwels, G. Peeters, "Evaluating automatically estimated chord sequences," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, 2013, pp. 749-753.
- [14] Jason Brownlee, How to Prepare Univariate Time Series Data for Long Short-Term Memory Networks, <https://machinelearningmastery.com/prepare-univariate-time-series-data-long-short-term-memory-networks/>
- [15] Librosa library glossary <https://librosa.github.io/librosa/glossary.html>
- [16] V. Lostanlen, C.-E. Cella. Deep convolutional networks on the pitch spiral for music instrument recognition. In Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR) 2016
- [17] Sergey Ioffe, Christian Szegedy - Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015
- [18] Stefan Gasser, Franz Strasser, SG1, Mirex 2018 submission
- [19] Mirex Results – https://www.music-ir.org/mirex/wiki/2018:Audio_Chord_Estimation_Results
- [20] Taemin Cho dataset <https://github.com/tmc323/Chord-Annotations>
- [21] Isophonics dataset. Centre for Digital Music (C4DM) at Queen Mary, University of London
- [22] Jason Brownlee – Train neural networks with noise to reduce overfitting -- <https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/>

[23] Jason Brownlee in Deep Learning for Computer Vision - How Do Convolutional Layers Work in Deep Learning Neural Networks? - <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>