THIRD YEAR PROJECT REPORT

3YP Title Here

Authors:

Kitty Fung, Edward Gunn,

Terence Tan, Di Wan

Supervisors:

David De Roure, Kevin Page

April 15, 2022

UNIVERSITY OF OXFORD

Abstract

Faculty Name

Department of Engineering Science

Third Year Project

3YP Title Here

by Kitty Fung, Edward Gunn, Terence Tan, Di Wan

The Project Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Contents

List of Figures

List of Tables

Chapter 1

Project Definition

1.1 Introduction

The process of writing music often begins with and idea for a melody which is followed by the development of chords to accompany it. The matching of chords to a melody is a skill which usually requires years of training in musical techniques such as harminisation. There is a large market of amateur musicians who lack the necessary training for this task but would otherwise enjoy the experience of developing music. In this report we will detail the design of **INSERT NAME HERE**, a system which facilitates the generation of an appropriately matched set of chords to a given monophonic melody. Users are able to record a melody using their microphone and regenerate the chord sequence until they feel the they have found one suitable for the intended feel of their song. Songs and generated chord accompaniments can be played back to the user, saved to a library of songs and shared using our songwriting community feature.
The problem of converting a recorded melody to a set of chords can be decomposed into a set of simpler sub-probblems, these being: The conversion of the recorded melody into a form which numerically represents its features. The generation of a set of chords from this numerical representation. The latter of these two problems can be solved in many ways many of which require a detailed knowlege of music to find patterns in melodies to match to chords. **WHY IS THIS A PROBELM FOR US** A method which requires little knowlege of music is the used of a machine learning model to extract these patters from data of known chord melody pairings from professionally composed music. The curation and processing of an appropriate dataset

requires significant effort and care in itself. This leads to a natural division of labour across the project into the catagories of: User Interface and general product design - detailed in **??** by Di Wan; Curation and preparation of a dataset in an appropriate format - detailed in **??** by Terence Tan; The design of an appropriate machine learning model - detailed in **??** by Edward Gunn; The conversion of recorded melody to the same format as expressed in the dataset - detailed in **??** by Kitty Fung.

1.1.1 Musical terminology

Throught this report we will use a variety of musical terminoligy which will be defined here. A piece of music is composed of a sequence of adjacent **measures**, periods of time in which notes and chords can be played. We will generally take a **measure** to mean a bar in the music, however it is not restricted to this. We restrict **chord** to refer to a triad of notes, the justification for this is explained in **reference to explanation**. The use of **melody** refers to a monophonic melody in which only one note is played at a time, excluding accompanying chords, unless otherwise stated.

1.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

1.2 Project management

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

1.3 Tech strat

1.4 Risk

1.5 Finance

1.6 Ethics

From the National Society of Professional Engineers Code of Ethics **codeofethics**, Engineering work impacts life directly, thus it is of the utmost importance to ensure service upholds honesty, impartiality, fairness, and equity, and must be dedicated to the protection of the public health, safety, and welfare. Since our design interacts with users through a mobile application, ethical concerns have arisen regarding the interactions between users as well as data handling on our end. In this section we will address issues related to data ethics and online hate crime.

1.6.1 Data ethics

Governmental organisations have published regulations **EUdataregulations2018** and frameworks **framework** that are minimum standards for handling data ethically. The core values of data ethics is to treat data responsibly and righteously.
There are 3 overarching principles**framework** to adhere to but in short, we as the developer should treat our users, data as we would for our own.

Transparency

We should be clear and publish specifications of our project in a way that users can understand and access easily. It is necessary to delineate a privacy policy and makes sure users read and agree to it before they use our application and keep the policy at a prominent location, i.e. in the footer for webpage, in the developer section of the app listing or on the sign-in interface of the app.
We would include the following items in our privacy policy:

1. Company licence and registration reference number: Registration at the Information Commissioner's Office is required before collecting data from the public

2. Details and purpose of data collected: We should limit data collection to what is necessary and to an explicit purpose. Thus, we will collect

   Personal details - Name and email address are the necessary fields, gender, age and profile picture are optional data that will help improve user experience and accuracy of the model.

   Audio data - Audio signal and metadata to feed in our machine learning model to improve accuracy.

   Networks and connections - for our interactive community features

   Usage - Contents and functions that the users have viewed or engaged with, as well as the duration.

   Device information - Device attributes like signal strength, battery levels, version of operating system

   Cookie data - Both first- and third-party cookies are needed. For the first-party cookies, to personalise and optimise user experience, we would save their usage preference like language, dark or light background mode. There are 2 purposes for using third-party cookies. Firstly, since we will be displaying advertisements, these advertising companies will be able to place cookies on our app for standard users, but not the premium users who subscribed to our services. Secondly, we will be implementing social sign-on so users do not have to create a new account on our end. In this case, the social media platform will be placing their third-party cookies on our app.

It is important to note that users have the right to know the third parties that have access to their data.

3. Data retention policy: We will specify how long information will be kept and the procedure of disposing the data when an user deactivated his account or when the data is no longer necessary to collect. Moreover, we would have to erase or rectify inaccurate data without delay. Although there is no limitation on data storage, we will have to act ethically and decide the timeframe based on the genuine motivation of retaining the data. Data shtould only be collected when it is vital in the context of app operation, thus we should not keep data just in case it is needed in the future. For data that has expired (past the retention period), we would have to either delete it or anonymise it. If we are to delete the data, we have to ensure all digital and hard copies of the data are destroyed. This action requires careful documentation of data storage from the date we collect data from users as traces may often reside in forgotten databases. Anonymising data means that a piece of information cannot be associated with an identity, which will not help with improving user experience, but still can be used to monitor the entire application performance.

4. Access rights within the team: Different team members of our project will have access rights to different types of data collected from the users, and we should delineate who will be responsible for which part of the data we store.

5. Rights that users have over their data: Users should have the right to request a copy of data provided, request us to delete the data and object to our data processing.

6. Notification of changes to privacy policy: We will contact users to review and accept the revised policy through email.

7. Security standards: We should specify the encryption standards and the processes in place to test the confidentiality, integrity and availability of our system.

8. Contact information: Organisation contact number, email address, office and postal address

Accountability

We should process personal data responsibly and systematically, as well as endeavour to reduce risks for individuals and mitigate social and ethical implications.**principles** This can be done by creating a department that overviews the entire project and ensures data is managed ethically throughout the process. As we have an interactive community feature, we must ensure that our app does no harm in any way. A new Online Safety Bill is being drafted recently to fight online hate crime, and online companies are liable for failures to deal with inappropriate material postsed online.**francis parliamentlaw** Companies have the responsibilities to confine illegal and harmful discussions and a conventional and efficient method to guarantee this is to implement algorithmic censorship.
Moreover, to be publicly accountable, effective governance and oversight mechanisms that can be exercised by the public are necessary. We can enact this by hiring an independent third-party audit review our data processing.

Fairness

When processing data, we should ensure no societal, racial or health bias is involved. Our project collects minimal personal data so the problem of differential processing for distinct groups can be avoided. On the other hand, another perspective of fairness can be manifested in the form of **Responsible Research and Innovation (RRI)**:

RRI is a process that seeks to promote creativity and opportunities for science and innovation that are socially desirable and undertaken in the public interest.**ukri** Not only does research bring novelty and value to society, it may also bring forward ethical dilemmas, social transformations and adverse consequences to society. Thus it is key to put emphasis on innovating responsibly and creating changes that have positive societal and environmental impacts.
There are a few areas of RRI that we should keep in mind:

1. Anticipation: We should vision the consequences of our research and innovation conducted. It is to ensure that the consequences of undertaking the research are considered and reflected in the research design. Although in this project we are not creating physical technology, we are building a platform that allows users to communicate with each other and if not dealt carefully, our app may become a breeding ground for online hate crime.

2. Reflection: researchers should always reflect on the research question they are investigating, the type of data collected, the method used to analyse the data and the implications of the findings. They should also judge if the research is required. An organisation can force reflection through organisational processes and structures like a project advisory board or quality assurance reviews. Reflection allows researchers to review the project from a macroscopic point of view.

3. Ethics: Researchers should uphold integrity and prevent misconduct. They should take accountability for both the undergoing research and behaviour.

4. Gender Equality: Since male engineers still dominate the engineering industry, it is easy to be biased in a project design, i.e. building models and interfaces that better suit male users. Thus it is paramount to have opinions from the underrepresented group as to create a comprehensive innovation.

5. Open Access: Making the research output publicly available to everyone so the whole society can be benefitted by reducing wasteful duplication, increasing transparency and reproducibility of results.

6. Governance: Organisations should establish practices that foster RRI, i.e.

   - Having transparent and reflective internal procedures
   - Promoting participatory governance
   - Fostering stakeholder engagement exercises
   - Encouraging future-oriented governance
   - Valuing responsiveness

7. Public Engagement: Researchers should settle upon a motivation and suitable audience before any public engagement. For our project, the goal would be to allow amateurs to enjoy composing music without music professionalism.

1.7 Sustainability

As technology advances, energy consumed by computers and digital personal devices is taking up a larger portion in worldwide energy consumption. There are about 8,918,157,500 active mobile devices consuming about 2 kWh energy per year**energy**. While a lot of effort has been put into reducing the energy consumption from the hardware perspective, it is also prime to focus on the impact that software implementation has on the energy consumption of a program. Other than writing energy-efficient codes (i.e. optimised searching and sorting algorithms), the programming language chosen can make a substantial difference too. Energy, time and memory size are classified as the major resources required for running a program. At first glance, since $Energy = Power * Time$, there seems to be a correlation between energy and time. Yet, Pereira et al. 2017**energyplanguage** suggests that since *Power* is not a constant, we cannot draw the conclusion that when *Time* increases, *Energy* must increase. Pereira ranked the performance of different programming languages according to its energy, time and memory size consumption (p.7). Surprisingly, there is no strong correlation between these 3 components. Instead, the performance depends on the category of the programming language. The programming languages can be divided into 3 execution types: compiler, interpreter and virtual machine.

Compiler converts the entire high-level language code to machine-readable language all at once. Some examples are C, C++, Rust and Go.

Interpreter translates one statement of the high-level language code to machine-readable language at a time. Common interpreted languages are PHP, Python and JavaScript. Thus, even if an interpreter takes less time to analyze the source code, it takes longer time to execute the process.

Virtual machine executes an intermediate code translated from the high-level language inputted. Some of the popular languages that uses virtual machines are Java, Scala, JRuby.

Since our application has no requirements on the computational time and memory size, we can solely focus on the energy consumption between different languages. Even though the energy needed to run different algorithms varies for different languages, it is certain that compiled language requires less energy compared to virtual machine and interpreted languages thus it would make sense to use C/ C++/ Rust to code.

Also, since we plan to employ a cloud network architecture and host most of the operation on cloud, we would be creating an enormous carbon footprint. Carbon3IT estimates datacentres account for at least 12% of UK electricity consumption, which is equivalent to 41.11TWh a year **carbon** Therefore, we should not underestimate the

1.8 Legality

Chapter 2

Product Design

2.1 Main Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

2.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

2.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor.

Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

2.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Chapter 3

Data

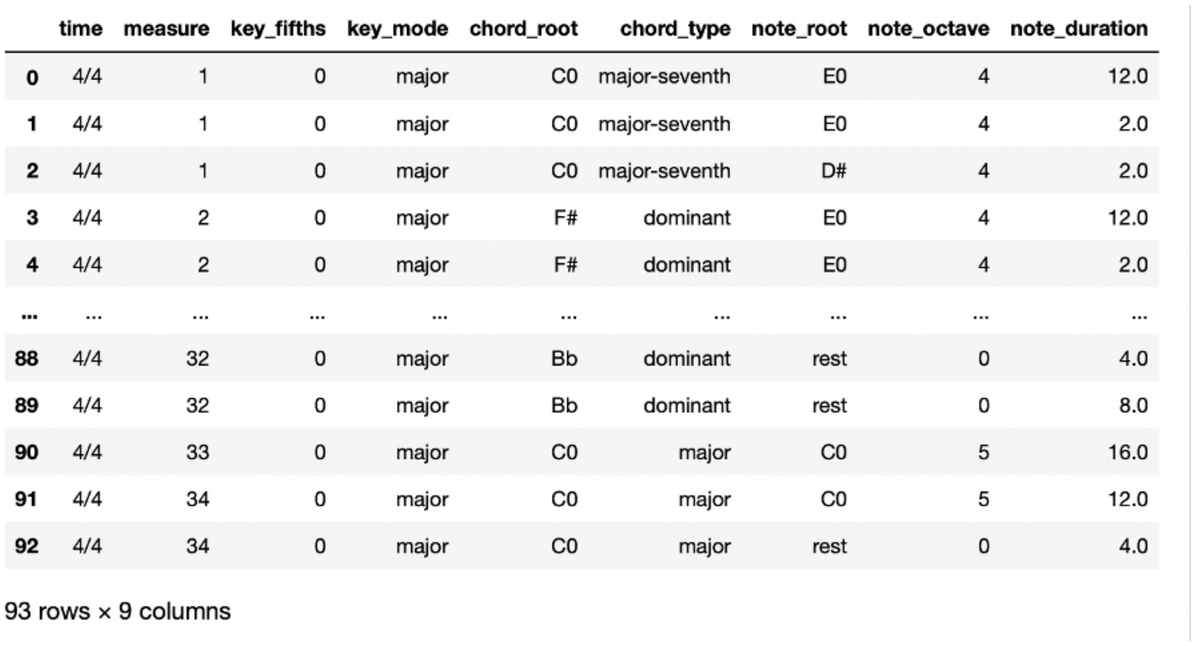
3.1 Dataset

The dataset used to train and test the machine learning model was obtained from another paper that was working on a similar project. The authors of that paper made their dataset available online. There are 2252 songs in this dataset, 1802 of which had been categorised as the training set while the rest had been categorised as the test set. Each song is in major key and only have a single chord per bar. For each song, all the relevant features had been extracted and placed into a single *CSV* file. These files can then be read and converted to DataFrame format using Python *Pandas* as shown in Figure **??**.

As can be seen in Figure **??**, the rows each contain information about a single note. Each bar is taken to be a single measure. The columns each represent a different piece of information about that particular note. *time* refers to the time signature, *measure* refers to the measure to which that particular note belongs to, *key_fifths* indicates the number of sharps/flats (e.g. -1 for one flat and 1 for one sharp), *chord_root* is the root of the chord with *chord_type* indicating the type of chord, *note_root* identifies the particular single note of that row, *note_octave* is the octave of that note, and *note_duration* indicated the duration of the note (4.0 for a quarter note).

3.2 Preprocessing of dataset

The dataset has to be preprocessed in order to make things simpler later on.



	time	measure	key_fifths	key_mode	chord_root	chord_type	note_root	note_octave	note_duration
0	4/4	1	0	major	C0	major-seventh	E0	4	12.0
1	4/4	1	0	major	C0	major-seventh	E0	4	2.0
2	4/4	1	0	major	C0	major-seventh	D#	4	2.0
3	4/4	2	0	major	F#	dominant	E0	4	12.0
4	4/4	2	0	major	F#	dominant	E0	4	2.0
...	...	...	...	...	...	...	...	...	...
88	4/4	32	0	major	Bb	dominant	rest	0	4.0
89	4/4	32	0	major	Bb	dominant	rest	0	8.0
90	4/4	33	0	major	C0	major	C0	5	16.0
91	4/4	34	0	major	C0	major	C0	5	12.0
92	4/4	34	0	major	C0	major	rest	0	4.0

93 rows × 9 columns

FIGURE 3.1: A song in DataFrame format after being read from CSV file

1. All songs are transposed to C major key. The key of a song determines the notes and the set of chords present in the song. Transposing all songs to a common key will basically normalise the different features of melodies and chords in different songs. The number of chord types present in the dataset will be reduced, which will decrease the number of chord types during the training process. Each song can be shifted to a different key without loss of the song's subjective character by shifting all the pitches equally.

2. The time signatures are all normalised. Different songs have different time signatures. To do so, each *note_duration* is multiplied by the reciprocal of the time signature *time* to give a normalised note duration.

3. Chord types are restricted to major and minor chords. All other chord types are converted to their most similar major or minor chords.

4. Some measures in the dataset contain rest notes. These measures are removed from the dataset.

5. Octave information is not required and is removed from the dataset.

6. There are also some irregular notes present in the dataset such as ′B-2′ and ′A2′. The numbers after the letters do not seem to represent octave information and the paper from which this dataset was obtained made no mention of them. Given that they represent a very small portion of the dataset, measures containing these irregular notes are also removed.

Using *Pandas* to remove the unwanted measures mentioned above and to normalise the note durations is a straightforward task. However, shifting all the songs to C major key is trickier. We would need to know the original key of the song, and then transposed the *note_root* and *chord_root* appropriately to C major key. The original keys of the songs are stated implicitly by their *key_fifths*; since we know all the songs are in major key and that each major key has a unique number of sharps/flats, the numberical value of *key_fifths* can be mapped to a specific major key as shown in Table **??**. Note that there exist more values of *key_fifths* than shown, but preliminary analysis of the dataset shows that only integer values of *key_fifths* from -6 to 7 are present within it. We also create a mapping of the 12 notes to a numerical representation as shown in Table **??** to make the processing easier later.

Using Table **??** & **??**, we can list all the major keys present in the dataset and convert the pitches that exist within each major key into their numerical representations (which goes from 1 to 12 and then loops back to 1) as shown in Table **??**. As expected, the differences between the pitches of the same key are consistent across all the major keys (e.g. the difference between Pitch 1 and Pitch 3 is always 4 for every major key), which shows that we can indeed tranpose a song to a different key by just shifting all the pitches equally. For each DataFrame row, we just have to convert *key_fifths* to the corresponding major key using **??** to obtain the numerical representaion of Pitch 1 of that major key. We also convert *note_root* and *chord_root* to numbers using Table **??**. Next, the numerical representation of Pitch 1 is subtracted from those of *note_root* and *chord_root*, and the differences are added to Pitch 1 of the C major key (which is 1) to obtain the shifted pitches in notes and chords respectively. Note that the differences may be negative, which would lead to a shifted note/chord that is outside of the 1-12 range. This is easily rectified by using *if* statements to check if the shifted note/chord is non-positive and to add 12 (since a zero would loop back to 12) to it if

TABLE 3.1: Mapping of *key_fifths* to major key

key_fifths	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
Major key	Gb	Db	Ab	Eb	Bb	F	C	G	D	A	E	B	F#	C#

TABLE 3.2: Mapping of music notes to numerical representations

C/B#	C#/Db	D	D#/Eb	E/Fb	F/E#	F#/Gb	G	G#/Ab	A	A#/Bb	B/Cb
1	2	3	4	5	6	7	8	9	10	11	12

so.

The next step is to convert all the chord types to either major or minor chords using the mapping shown in Table **??**. This mapping has been checked by the supervisors of this project and deemed to be reasonable. Do also note that Table **??** does not contain an exhaustive list of all chords in existence, but only those that were found to be present within the dataset.

3.2.1 Code

As mentioned above, we first use *Pandas* to clean up the data. A nested loop can then used to loop through each DataFrame row for each song. The steps outlined above can then be applied to each row. At the end of the nested loop, the dataset is now fully preprocessed.

3.3 Model input format

Now that the dataset has been preprocessed, it can now be converted into a format that is appropriate for input to the machine learning model.

3.3.1 LSTM format

The LSTM data input format is a matrix with 37 columns, with each row containing informaion about a single measure. The first 12 columns represent the 12 note (C,C#, etc.), the next 24 columns represent the 12 major chords and the 12 minor chords, and the last column correspond to the absence of a chord. If a particular note exists within a particular measure, the element that corresponds to that particular measure

TABLE 3.3: The component notes/pitches of each major key

Major key	Pitch 1	Pitch 2	Pitch 3	Pitch 4	Pitch 5	Pitch 6	Pitch 7
C#	2	4	6	7	9	11	1
F#	7	9	11	12	2	4	6
B	12	2	4	5	7	9	11
E	5	7	9	10	12	2	4
A	10	12	2	3	5	7	9
D	3	5	7	8	10	12	2
G	8	10	12	1	3	5	7
C	1	3	5	6	8	10	12
F	6	8	10	11	1	3	5
Bb	11	1	3	4	6	8	10
Eb	4	6	8	9	11	1	3
Ab	9	11	1	2	4	6	8
Db	2	4	6	7	9	11	1
Gb	7	9	11	12	2	4	6

TABLE 3.4: Mapping of chords present within the dataset to major/minor chord.

Major	Minor
Dominant-ninth	Minor-seventh
Major-sixth	Minor-sixth
Major-seventh	Diminished
Dominant	Half-diminished
Suspended-fourth	Minor-ninth
Augmented-seventh	Diminished-seventh
Major-ninth	Minor-eleventh
Dominant-seventh	Minor-major
Augmented	Major-minor
Dominant-thirteenth	Minor-thirteenth
Power	Minor seven flat five
Suspended-second	
Dominant-eleventh	
Pedal	
Major 6/9	
Augmented-ninth	
Sixth	

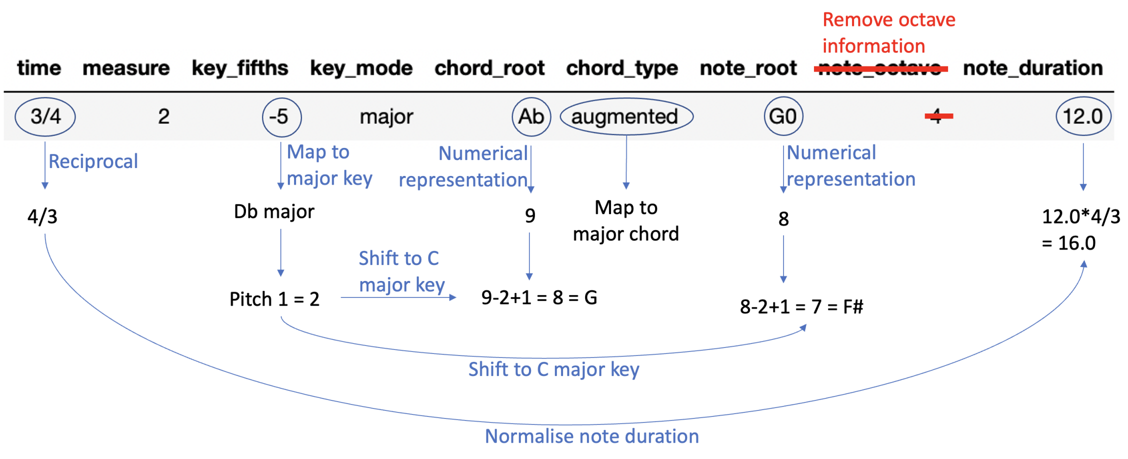


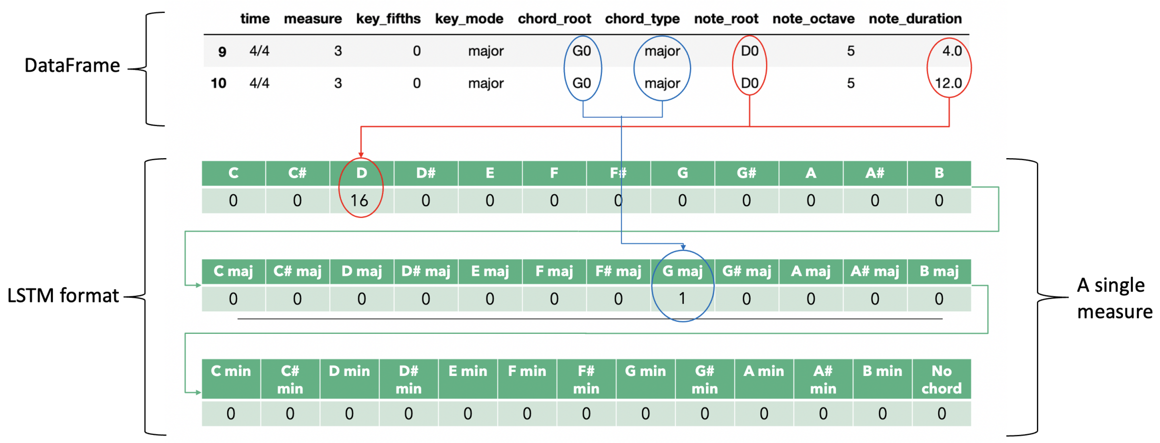
FIGURE 3.2: Pictorial representation of the preprocessing of the dataset.

(row) and particular note (column) will be the normalised duration of that note. Similarly, the element that corresponds to the chord of a particular measure will be a '1'. Notes and chords that are not present within that measure will be marked with a '0'. If no chords are present within a measure, the last column will be marked as a '1'. A pictorial representation of the transformation is shown in Figure **??**.

Code

We again use a nested loop to loop through each DataFrame row for each song. Two 1 by 37 arrays *LSTM_data* and *new_row* are initialised. As we loop through each row, we keep track of the meaasure. As long as the measure does not change, *new_row* is progressively updated with the normalised note durations of the present notes in this manner: $new\_row[0, note-1] = new\_row[0, note-1] + normalised\_note\_duration$, where *note* is the numerical representation of the note present in each DataFrame row and *normalised_note_duration* is the normalised note duration of that note. This works because the mapping between the notes and their numerial representation starts at 1 and ends at 12 (as can be seem in Table **??**) and indexing starts at zero in Python. Hence, the (*note*-1)th column of the LSTM format (and thereby *new_row*) will correspond to *note*.

Once the measure changes, the now complete *new_row* for the previous measure is concatenated with *LSTM_row* along the row axis, i.e. *new_row* is added to the bottom of *LSTM_row*. The elements of *new_row* are reset to zero before *new_row* is



	time	measure	key_fifths	key_mode	chord_root	chord_type	note_root	note_octave	note_duration
9	4/4	3	0	major	G0	major	D0	5	4.0
10	4/4	3	0	major	G0	major	D0	5	12.0

C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	16	0	0	0	0	0	0	0	0	0

C maj	C# maj	D maj	D# maj	E maj	F maj	F# maj	G maj	G# maj	A maj	A# maj	B maj
0	0	0	0	0	0	0	1	0	0	0	0

C min	C# min	D min	D# min	E min	F min	F# min	G min	G# min	A min	A# min	B min	No chord
0	0	0	0	0	0	0	0	0	0	0	0	0

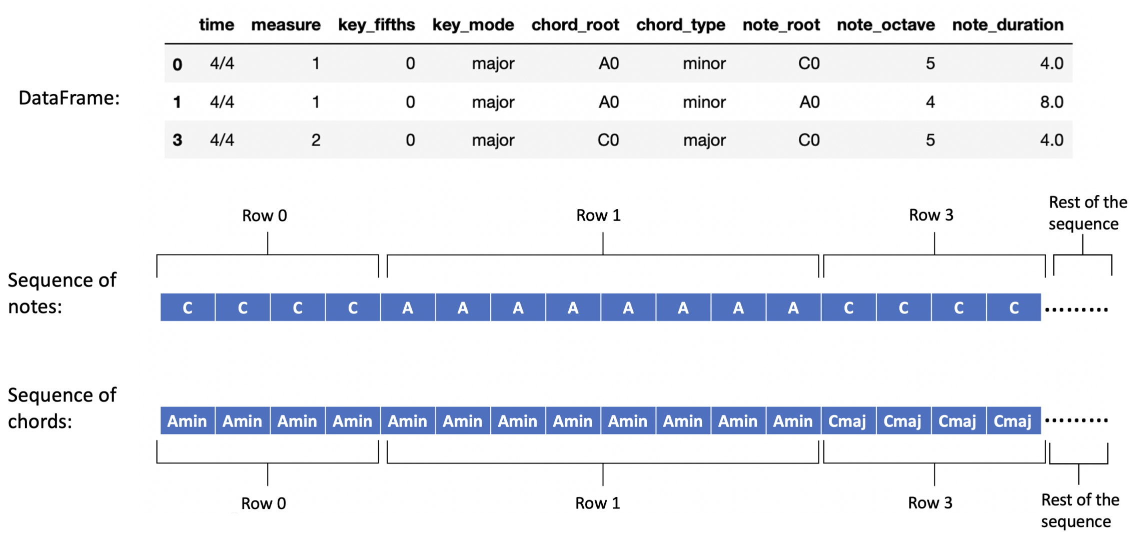
FIGURE 3.3: Transforming DataFrame to LSTM data input format.

updated with the note and chord information of the current measure. The note information can be updated as explained earlier. For the chord information, we can use *if* statements to set $new\_row[0, chord + 11] = 1$ if $chordtype$ = ′major′, or to set $new\_row[0, chord + 23] = 1$ if $chordtype$ = ′minor′, or to set $new\_row[0, -1] = 1$ otherwise (no chord present), where *chord* is the numerical representation of *chord_root* (mapped using Table **??**). Since we know that there is only one chord type per measure, we only have to update the chord information for *new_row* once, at the start of a new measure. The start of the first measure can be taken to be a special case of a measure change (in this case, transition from a measure initialised as ′unknown′ to the first measure of the DataFrame).

3.3.2 Transformer format

The transformer requires two data inputs: a single sequence of notes for each song, and a separate sequence of chords for eaach song. Both sequences are constructed by going down the Dataframe rows, and adding the notes/chords of each row to the respective sequences based on the *note_duration* value, e.g. a sequence of 4 ′C#′s for a ′C#′ with a *note_duration* of 4.0, and a sequence of 16 ′Bmaj′ for a ′B major′ of *note_duration* of 16.0. Another example is presented in Figure **??**. Of course, this means that only integer values of *note_duration* are accepted and measures with non-integer values have to be removed beforehand.

It is obvious that the sequences will be very long given that just the three rows in



DataFrame:

	time	measure	key_fifths	key_mode	chord_root	chord_type	note_root	note_octave	note_duration
0	4/4	1	0	major	A0	minor	C0	5	4.0
1	4/4	1	0	major	A0	minor	A0	4	8.0
3	4/4	2	0	major	C0	major	C0	5	4.0

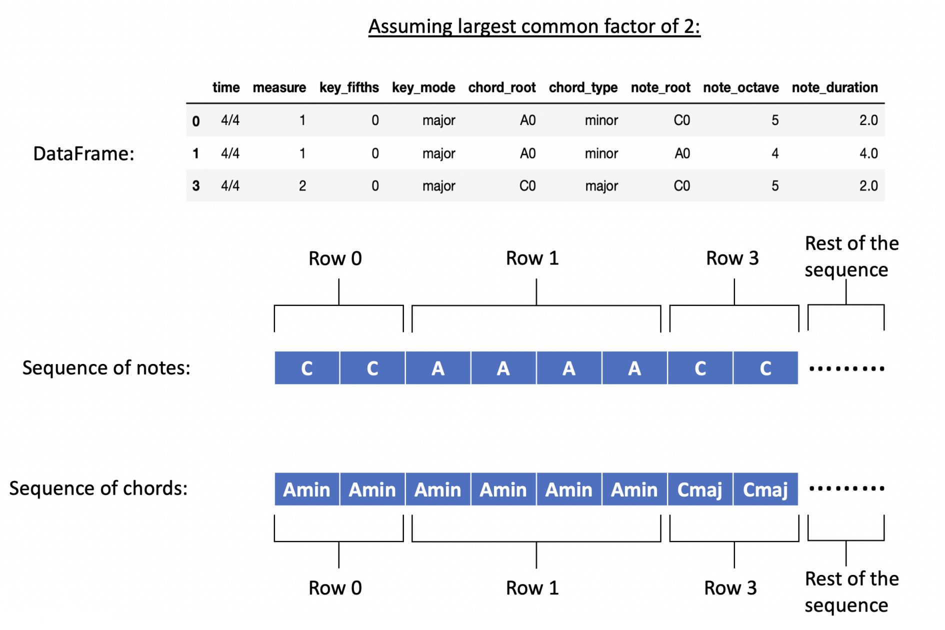
Row 0
Row 1
Row 3
Rest of the sequence
Sequence of notes:
C C C C A A A A A A A A C C C C ....
Sequence of chords:
Amin Amin Amin Amin Amin Amin Amin Amin Amin Amin Amin Amin Cmaj Cmaj Cmaj Cmaj ....
Row 0
Row 1
Row 3
Rest of the sequence

FIGURE 3.4: Converting DataFrame to Transformer data input format.

Figure **??** resulted in 16 elements for each sequence. As shown in Chapter 4, the time complexity of the Transformer is n^2. Hence, it is crucial to reduce the sequence length to shorten the training time. This can be achieved by dividing all *note_duration* by their largest common factor for all measures within a single song. This may not reduce the length of the sequences for all songs (songs with *note_duration of 1.0* will have a trivial largest common factor of 1.0), but it will still reduce the sequence length for some songs, as shown in Figure **??**, which will decrease the training time significantly.

Code

The same nested loop as before is used again for this. Four empty lists *note_duration_li*, *note_li*, *chord_li*, *chordtype_li* are initialised. As we loop through the DataFrame rows of a single song, we append the information from each row to the respective lists. At the end of this inner loop, the largest common factor of *note_duration_li* can be found using the *math.gcd* function. Since *math.gcd* only accepts two arguments, we initialise *lcf* as the index 0 element of *note_duration_li*, and loop from the index 1 element to the last element of *note_duration_li*. Within this loop, *math.gcd* takes in *lcf* and *note_duration_li*[*j*] as its two arguments, where *j* is the currect loop index. In essence, we are just finding the largest common factor of the first two elements of



	time	measure	key_fifths	key_mode	chord_root	chord_type	note_root	note_octave	note_duration
0	4/4	1	0	major	A0	minor	C0	5	2.0
1	4/4	1	0	major	A0	minor	A0	4	4.0
3	4/4	2	0	major	C0	major	C0	5	2.0

FIGURE 3.5: Converting DataFrame (normalised by the largest common factor) to Transformer data input format.

note_duration_li, and finding the largest common factor of the previous largest common factor and the third element, and so on. This will give us the largest common factor of all the values stored in *note_duration_li*, which are all then divided by this largest common factor to give *normalised_note_duration*.

We can now start to construct the sequences of notes and chords. Two empty arrays *row_note* and *row_chord* are initialised, and we loop through *normalised_note_duration* with loop index k. For the k^{th} element of *normalised_note_duration*, a 1 by *normalised_note_duration*[k] array filled with '1's is initialised and multiplied by *note_li*[k]. The result is concatenated with *row_note* along the column axis. For the chord sequence, a 1 by *normalised_note_duration*[k] array filled with '1's is also initialised. This array is multiplied by *chord_li*[k] if chordtype_li[k] is 'major', by $(chord\_li[k] + 12)$ if it is 'minor', and zero if there are no chords. The resulting array is then concatenated with *row_chord* along the column axis.

After looping through all the DataFrame rows of a song, *row_note* and *row_chord* are the now complete sequences of notes and chords respectively for that song. These will be the input to the Transformer model.

Chapter 4

Model

4.1 Introduction

The problem of generation of a set of chords from a melody is very similar to that of translating one language to another. The translation problem is one that is very popular in machine learning research and thus there is many resources on it. However the music related problem is harder to solve due to the extra dimension each of its elements contains. Each element in a melody has both pitch, represented by a descrete symbol or note, and duration whereas each element in the sequence of language is composed of only the discrete symbols or words. Therefore in order to use techniques developed for natural language processing it is necessary to encode the melody and chords in such a way that their dimensions are collapsed into one. Since this collapsing of dimensions has already been conducted in **REF TO DATA CHAPTER** we are able to take full advanatges of NLP techniques. There has been many attempts to apply some of these techniques to the chord generation problem. In order to find the best model we evaluate these attempts against a defined criteria and develop a novel model to evalutate against the same criteria.

4.2 Model Requirements

4.2.1 MVP Requrements

The MVP requirements were defined at the beginning of the project to ensure it is compatible with the other parts of the product that were simaltaniously in development and to ensure the possibbility of the creation of the prototye for proof of concept. We required the model to be a black box in which we could input a melody and it would output chords which sounded good. The notion of sounding good is intentionally vague as what sounds good or bad is usually down to individual taste in music. There is never a definitive answer to which chords would sound the best. However, we felt **NEED STRONGER PROOF** that even people without any musical training would be able to judge whether chords fit the melody relatively accurately. Within this overarching requirement we defined tighter constraints for the sake of both practicality and user experience. The model would have to be designed for sequential data such that the temporal relationship of the different notes being played were taken into account. It would be possible to simply learn a function where for a given measure a chord suited to the notes played within that measure was generated without taking into account surrounding measures. This approach could produce reasonable results and be significantly more simple that other options, however, the quality and variation of chords generated would be lower than that of a model for sequential data, hence our choice to forego it. The model would also have to be conditional. For a given input it should produce a catered output. This is an obious contraint however for models such as a GAN it recquires changes to the format of the input data. To maximise user experience the models should be non-deterministic. This allows users to regenerate the accompaniment multiple times to obtain new chords and thus means they can choose what they feel is best. For practicality sake we constrained our MVP to only require one chord to be played each measure as the problem of determining when a chord should be played is of similar difficulty to choosing which chord to play.

4.2.2 Other possible Requrements

There were some constraints which were not used in our project which would provide better quality chords but provided too large a practical barrier to be implemented. The act of collapsing the pitch and duration of the notes into a single dimension removes information from the melody. In our case this information is the order in which the notes are played and the rhythmic intention of the composer. It is likely a model would be able to produce more suitable chords given this information. Thus, for a stricter set of constrains we could include the requirement that the order and duration of notes are both maintained in the data. The accompaniment for music is not limited to a single chord played at the start of each bar. A much more interesting accompaniment would be generated if it were not limited to this restrictive format. As shown in **ReinforcementLearning** it is possible to create a model which learns the divisons within the music and thus learns when would be most appropriate to play a chord. Therfore, in order to allow for more interesting accompaniment, the requirement of one chord per bar could be changed such that chords should be played with closer freedom to that of a human composer.

4.2.3 Evaluation criteria

Approach Many attempts to solve the chord generation problem have been made each of which containing variants on models and implementations which result in large variations in mertrics used for evaluation. There is also a large variation in features for which quantative evaluation is impossible. As it would be impractical to implement each relevent model ourselves to allow for full standardistation of evaluation we will evaluate models built ourselves and from previous work based on a set of criteria defined below.

Data One of the biggest influences on the effectiveness of a model is the dataset on which it is trained. In general a larger dataset results in a better generalisation of learning **Need proof**. Most datasets are comprised of lead sheets which can be translated into chord melody pairs. Many datasets are then further divided down into measures in which there is one chord played per measure and the melody within that measure is somehow encoded. Thus for best comparison the number of measures used in the dataset is a good criterion for evaluation. This is a particularly important area for evaluation as it will strongly affect the output of the model and thus the results of other

Quantiative Evalutaion There is difficulty in quantative evaluation for this problem as there is no strictly correct chords for a given melody and thus comparison to any specific set of chords gives a skewed interpretation of the output. However, the use of conventional quantative evaluation does still correlate strongly with the sentiment given by a human judging the quality of chords and thus can be cautionsly used in evalution. The test set from the data can be used to compare the outputs from the model to previously assigned chords. Accuracy can be found by finding the number of chords successfully predicted and dividing by the total number of chords produced.

$$\text{Accuracy} = \frac{\text{Number of Chords Correctly Assigned}}{\text{Total Number of Chords Generated}} \tag{4.1}$$

As this was the only quantative measure consistently used across previous implementations this is the only one we will consider for evaluation.

Qualitative Evaluation Some previous work **MySong**, **BLSTM** carried out experiments to judge the sentiment of untrained musicicans to the generated chords. Participants were played the melody accompanied by a varying set of chords and asked to judge which chords they felt were best out of a set containing chords generated by models and some human written accompaniment. The results from these experiements can be used to compare models to each other as well as evlauate them relative to the standard set by human written chords. As well as evaluation based on the quality of chords produced we will discuss extra functionality made possible by some models and the effect this has on other evaluation metrics.

Criteria The final criteria used for evaluation are thus:

- Number of measures in the dataset
- Accuracy in testing
- Human sentiment towards generated chords
- Extra functionality the model provides

4.3 Models

The generation of chords to accompany a monphonic melody is an area which has developed as machine learning techniques are improved. An early approach to

this problem from **ChordPrediction** used a standard MLP to learn the relationship between melodies and accompanying chords. MySong, the first attempt focusing specifically on a vocal melody in **MySong** used an augment hidden markov model with parameters such that users could adjust the "Happy factor" and "Jazz factor" to alter the mood of the generated chords to their preference. A weakness of the hidden markov model is that at each timestep only the previous timestep is considers when suggesting a chord. **BLSTM** utilises a BLSTM to increase the effect of long term dependencies on the output rather than relying only on the previous output. **MLForChords** tested and evaluated a number of more simple models such a Logistic Regression, Naive Bayes, SVM and Random Forest trained on data from 43 lead sheets
ReinforcementLearning proposed a model that learns a structured representation for use in symbolic melody harmonization. It utilises two layers of LSTMs to detemine when each chord should be played then utilise a policy gradient RL method to select each chord. MuseGAN
ChordGAN
CLSTMGAN for melody Generation

4.3.1 Model evaluation template

Model explanation/history - why is it generally applicable

Previous implementaitions

General backgorund

Evaluation

Data

Quantative

Qualitative

Extra features

4.3.2 HMMs

Explanation and Applicability

Hidden Markov Models or HMMs were first put forward by Leonard E. Baum in a series of statistical papers **list papers from wikipedia**. They model a stochasitc process in which the desired states, $\boldsymbol{X}$, are not directly observable but related states, $\boldsymbol{Y}$, which directly influence the desired states are. By modeling the $\boldsymbol{X}$ and $\boldsymbol{Y}$ Markov Processes **references?** we are able to infer the hidden state. To apply an HMM to our problem we take the hidden state, $\boldsymbol{X}$, to be the chords played, and the observable state $\boldsymbol{Y}$ to be the Melody. It is notable that the conditional probability distribution of the hidden variable $x(t)$ at time t, only depends on that of the previous time step $x(t-1)$ and that $y(t)$ only depends on the hidden distribution at the current time step $x(t)$. For our puroses this limits the "memory" the model could have to a single measure and thus seriously reduces the effect of relationship between chords across time.

Implementations

MySong The first application of an HMM to this problem while also being the the first attempt focusing specifically on a vocal melody from **MySong** used an augmented hidden markov model with parameters such that users could adjust the "Happy factor" and "Jazz factor" to alter the mood of the generated chords to their preference. No measure was given for the accuracy of the model, however a study which compared the quality of MySong to Manually assigned chords was carried out. In 264 comparisons the participants prefered MySon 95 times, manual 121 times and had no preference 48 times The additional user input possible with the "Happy factor" and "Jazz factor" parameters are reported to significantly improve the user experience, however, this is thier only implementation and so nothing can be infered about the quality of the HMM itself.

Chord Generation from Symbolic Melody As a comparison for the main model in **BLSTM** an HMM was implemented. The HMM is trained on a reduced version of the **Include link?** wikifonia.org dataset which includes and array of Western music genres with 2,252 lead sheets, 1802 of which are used for training. This overall comes to 72,418 measures for the training set and 17,768 for the test set. They tested the accuracy of the HMM on sequences of length 4, 8, 12, and 16 bars long gaining an accuracy of 0.4033, 0.4043, 0.4041, and 0.4045 respectively and an average accuracy of 0.4041. They also carried out a user subjective test with 25 participants each evaluating 18 sets, each set containing a melody acompanied by three generated chord sequences and an original chord sequence. The users would listen to the music played and judge the accompaniment on a scale of 1, being not appropriate, and 5, being very appropriate. The HMM achieved an average score of 2.31 and the original chords achieved an average score of 4.04.

Machine Learning in Automatic Music Chords Generation An HMM was tested in **MLForChords** along with other simpler models. It was trained on 43 lead sheets, with 813 measures total. An accuracy of 0.4844 was achieved, however an choice of only 7 chords were used unlike the usual 24.

4.3.3 DNN-HMMs

Explanation and Applicability

A deep neural network HMM makes it possible to assume a posterior for the HMM using the output from the softmax output layer of the DNN. This allows for the posterior to be learned in training.

Implementations

Chord Generation from Symbolic Melody This was implemented much like the HMM from **BLSTM** mentioned above in **??**. The same dataset was used and an accuracy of 0.4502, 0.4482, 0.4495, and 0.4468 was obtained on the 4, 8, 12 and 16 bars respectively. This results in an average accuracy of 0.4487. On the user subjective test the DNN-HMM achieved a score of 2.48.

4.3.4 GANs

Explanation and Applicability

Generative Adversarial Netwoks or GANs were first proposed in the landmark paper **GANs**. Unlike most models GANs consist of two separate agents working against each other. The first model, the Generator, takes an input of random noise and outputs data which mimics the training data. The second model, the Discriminator, takes as input an example from the training data or an output from the Generator and outputs a value between 0 and 1 indicating whether it thinks the input is real or generated. The Binary Cross Entropy loss is used for the Discriminator avaraged across real and generate examples. The loss for the Generator is the log complement of that for the Discriminator. Thus the two models play the following minmax game:

$$\min_D \max_G V(D,G) = \mathbf{E}_{x\sim p_{data}(\mathbf{x})}[logD(\mathbf{x})] + \mathbf{E}_{z\sim p_{data}(\mathbf{z})}[log(1-D(G(\mathbf{z})))] \tag{4.2}$$

Theoretically the Generator and Discriminator can be any differentaible function thus leaving much room for flexibility. The problem with GANs for our uses is that they are not conditional, the only input to the Generator is noise. A proposed remody for this was presented in **CGANs**. By concatinating the label data, in our case the melody, with the noise as input to the Generator and doing the same with the training data and the label data, in our case a chord and melody, as input to the Discriminator GANs can be made conditional. Thus they would be suitable for our uses.

Implementations

ChordGAN **ChordGAN** uses a conditional GAN along with Chroma Feature Extraction to generate chords for a specific genre of music based on the dataset it was trained on. They used three different data sets one for each Pop, Jazz and Classical music each shorter than 1600 measures (specific lengths are not given). The model achieved an acuracy of 0.68, 0.74 and 0.64 respectively across the datasets.

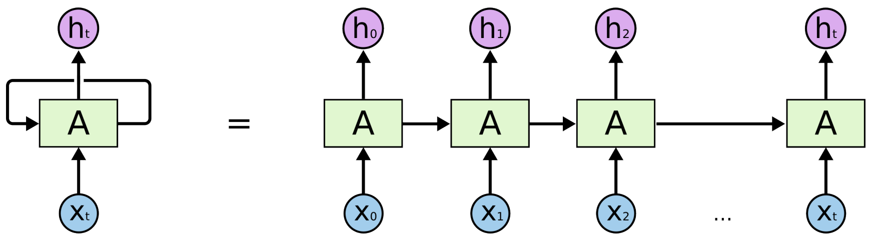


FIGURE 4.1: A many inputs to many outputs diagram of an RNN from **oinkina**

4.3.5 RNNs

Recurrent Neural Networks or RNNs have become a staple in the machine learning engineer's library of models. They are structured much like a normal MLP, however, they contain an extra connection to the state of the network in the timestep before. This means that the state of the network at each timestep depends that of the previous timestep and thus the state at the current timestep is affected by the state in all previous timesteps. This makes RNNs ideal for processing sequential data as temporal relationships are taken into account. The gradient of RNNs can be found using a variation of the backpropagation algorithm usually know as backpropagation through time. RNNs that operate on large sequences often experience the probblems of exploding or vanishing gradients leading to a saturation of learning.

4.3.6 LSTMs

The Long Short Term Memory model or LSTM was proposed in **LSTMs** in order to overcome the gradient problems related to RNNs and thus allow for faster training on long sequences. They are also capable or learning longer dependencies due to their internal memory. An LSTM cell has three gates: input, forget, and output. The state of these gates determines whether the cell allows new input, forgets old information, and affects the output at the current timestep. At timeset t, the states of the gates are given by:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{4.3}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{4.4}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{4.5}$$

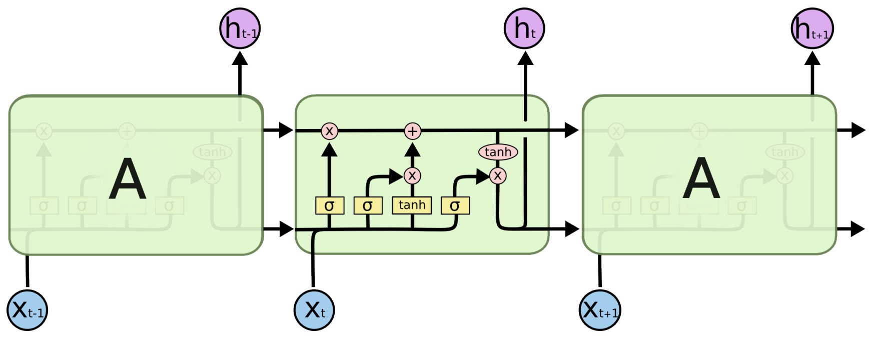


FIGURE 4.2: The repeating module in an LSTM from **oinkina**

where i_t, f_t and o_t denote the input, forget, and output gates state respectively, h_{t-1} is the output at the previous timestep. w and b represent weights and biases of each gate, x_t is the input to the LSTM cell, and $\sigma(.)$ is the sigmoid function applied elementwise. The current output of the cell is computed by:

$$h_t = o_t \circ tanh(c_t) \tag{4.6}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t \tag{4.7}$$

$$\widetilde{c}_t = tanh(w_c[h_t, x_t] + b_c) \tag{4.8}$$

Implementations

Chord Generation from Symbolic Melody This was the main model under scrutiny in **BLSTM** mentioned above in **??** and **??**. They build a bidirection LSTM with two hidden layers and used a hyperbolic tangent activation function. The softmax function is applied to the output in order to represent the probaility that each of the 24 chords is played. The same dataset as the HMM and DNN-HMM was used and an accuracy of 0.5055, 0.5032, 0.4923, and 0.4990 was obtained on the 4, 8, 12 and 16 bars respectively. This results in an average accuracy of 0.5000. On the user subjective test the DNN-HMM achieved a score of 3.55.

Automatic Melody Harmonization A particularly interesting model heaviliy relying on LSTMs surrounded by a reinforcement learning framework was proposed in **ReinforcementLearning**. The simultaniously trained a Structured Representation Module responsible for learning note-level, phrase-level and segment level representaitions, a Segmentation Module acting as a reinforcement learning agent to decide whether the current note is the boundary of a phrase or segment and a Harmonisation Module responsible for generating chords for each segment. They used the Hooktheory Lead Sheet Dataset with 10,000 songs, no number of measures is given but with the differnece in model architecture so drastic a comparison on this metric would be inappropriate anyway. The achieved an accuracy of 0.3742 and compared that to an SVM, CNN, LSTM and BLSTM with blocked Gibbs sampling which achieved an accuracy of 0.2516, 0.2664, 0.2802, 0.2933 respectively.

4.3.7 Transformers

Transformers have been a revolutionary development in the field of NLP, first proposed by **Transformers** they have become very widely used and regularly produce state of the art performance. Transformers utilise the encoder decoder model for sequence to sequence translation. They also heavily use attention which is a mechanism for weighting the importance of different elements of a sequence of data. Specifically they propose the Scaled dot-product attention shown in **??**

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{4.9}$$

where Q, K and V represent the queries, keys and values respectively and d_k is the dimension or queries and keys.

4.4 Our Model

We propose the use of a conditional BLSTM GAN to learn the association between melodies and chords. The Discriminator consists of a number of LSTM layers followed by a linear output layer. The Generator consists of a linear layer followed by a number of LSTM layers and then by another linear layer. The Binary Cross Entropy loss is used to determine the Discriminator loss for each example. This is averaged across every measure and then between real and generated examples

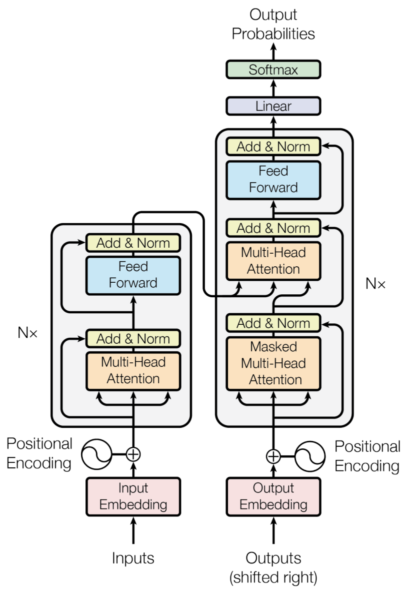


FIGURE 4.3: The architecture of a transformer from **Transformers**

4.4.1 Model Functionality

As proof of concept for the design we developed a command line based interface for the model such that it was easy to train, vary parameters and test. In production the user would be able utilise a subset of this interface, such as the option to generate accompaniment and have it played back, through a graphical user interface. The options available in the interface are show in table **??**

4.5 Training

Optimiser We used the Adam optimiser **Adam** which is recommended in the Stanford CS231n: Convolutional Neural Networks for Visual Recognition[1] as the default optimiser. **EXPLAIN ADAM WHEN NOT FALLING ASLEEP**

Loss As we are using a GAN we use the Binary Cross Entropy or BCE as our loss function

$$\frac{1}{N}\sum_{i=1}^{N} log(p(y_i)) \tag{4.10}$$

[1]https://cs231n.github.io/

TABLE 4.1: The parameters of the command line iterface for the model

Parameter	Options	Default	Description
`-input_size`	`[input_size]`	12	The size of the input vector to the discriminator representing the melody
`-output_size`	`[output_size]`	25	The size of the output vector representing the chord played in each measure
`-h_size`	`[h_size]`	128	The size of the hidden layers in the LSTM layers
`-n_layers`	`[n_layers]`	2	The number of LSTM layers in the generator and discriminator
`-noise_size`	`[noise_size]`	12	The size of the noise vector concatinated with the input vector to the generator
`-max_seqlen`	`[max_seqlen]`	200	The maximum length of song in measures used in training
`-src_data`	`[src_data]`		The default path to the training data
`-batch_size`	`[batch_size]`	10	The size of the batches used in the stochasitc gradient descent algorithm
`-epochs`	`[epochs]`	100	The number of epochs used in training
`-printevery`	`[printevery]`	10	The number of epochs between printing an example during training
`-load`		False	Whether to load a model in or not
`-load_dir`	`[load_dir]`		The path to the folder containing pretrained models
`-model_num`	`[model_num]`	1	The number of the model to be loaded in
`-save`		False	Whether to save the model after training
`-save_dir`	`[save_dir]`		The path to the save directory
`-playback`		False	Whether to play an example of generated music at the end of training

which for the disciminator takes the form

$$\frac{1}{N}\sum_{i=1}^{N} log(D(conc(x_i, z_i))) + log(1 - D(conc(G(z_i), z_i))) \tag{4.11}$$

where x_i is are the real examples, z_i are the melodies and the *conc* function concatinates the two vector arguments. The loss is the sum of the loss of the discriminator on real data and generated data. The generator loss is

$$-\frac{1}{N}\sum_{i=1}^{N} log(1 - D(conc(G(z_i), z_i))) \tag{4.12}$$

which results in tring to maximise the loss of the discriminator on generated data. This is equivalent to finding the discriminator loss on generated data when labeled as real which is the method we use in our implementation.

4.5.1 Avoiding Overfitting

Dropout GAN techniques Reducing number of LSTM layers

4.5.2 Issues

Training is conducted in batches of a given size. It is unlikely that the size of the dataset is divisble by the batch size so the final batch is likely to be shorter than the others. This initially casued problems, however, we were able to set the dimensions of relevant objects, such as the input noise to the generator, to be relative to the current batch size rather than the usual batch size thus solving the problem. Softmaxing of outputs of generator Device managment

4.5.3 Testing

4.6 Results

Chapter 5

Audio Processing

Before we feed the audio clip to our machine learning model, it is crucial to pre-process the signal as to achieve higher accuracy and avoid further deterioration. The choice and implementation of noise filter will then be explained in **LINK TO SECTION 1**. We then feed the filtered output to a pitch detection algorithm (PDA) **lINK TO SECTION 2** and then a key detection algorithm (KDA) **LINK TO SECTION 3** Figure **UPDATE!** shows the flowchart of the audio processing part of our project.

5.1 Assumptions

Before we delineate the approach to audio processing, there are some assumptions that our model is built on:

- **Assumption 1:** Users' audio input device does not contain active noise cancelling functions.

These assumptions will be referred to later on in the section.

5.2 Noise Filter

Noise filtering is essential as it reducess or eliminates the noise present in the input signal. A conventional method to quantify noise is to use signal-to-noise ratio (SNR), which is often represented in decibels.

$$SNR = 10 * log_1 0((P_{signal})/(P_{noise}))$$

As its name suggests, SNR is the power ratio between desired signal and undesired noise. Effectively, we would like to use noise filters to achieve a higher SNR.
There is a few sources of noise when an user record himself with a microphone. Firstly, there exists self-noise, which is the instrument noise produced by the microphone itself. Noise may be induced or created when the signal passes through electronic componenets like transistors and printed circuit boards.**selfnoise** The second source, ambient noise, contributes to a large portion of noise present in a recording. Room reflections, extraneous noise, electromagnetic interference and mechanical noise are some causes to the existence of ambient noise.

5.2.1 Possible Models

Most of the noise filters work in the frequency and spectral domain, here we are going to inspect and compare 3 noise reduction mechanisms.

1. Low-pass filter (LPF)

   LPF passes signals with $f < f_c$, where f_c is the cut-off frequency, and attenuates signals with $f > fc$.

   $$H(f) = rect(f/(2*B))$$

   $$h(t)) = \mathfrak{F}^{-}1H(f) = \int_{-B}^{B} e^(2(\pi)ift)\,df = 2Bsinc(2Bt)$$

   In order to implement an LPF, we have to transform signal from time domain to frequency domain using fourier transform. An ideal LPF would completely remove frequencies that are higher than f_c and is a non-ca)sual linear time-invariant system. The impulse response of an LPF is a sinc function that extends to [∞,-∞]. This is why it is impossible to realize an ideal LPF since that will take infinite time and memory.
   LPF avoids aliasing since it removes the high-frequency content but not the desired signal

2. Wavelet transform

Wavelet transform creates a representation of the signal in both time and frequency domain so localized information of the signal can be efficiently accessed. It is often compared with fourier transform (FT), which has the below limitations:

(a) For windowed FT, if the feature is larger or shorter than the window, it cannot be captured completely.

(b) Time resolution for high frequencies is the same for low frequencies. As frequency increases, rate of change of the signal increases, and high frequency signals contain more information in a window than that of low frequency, thus we need a higher time resolution for that.

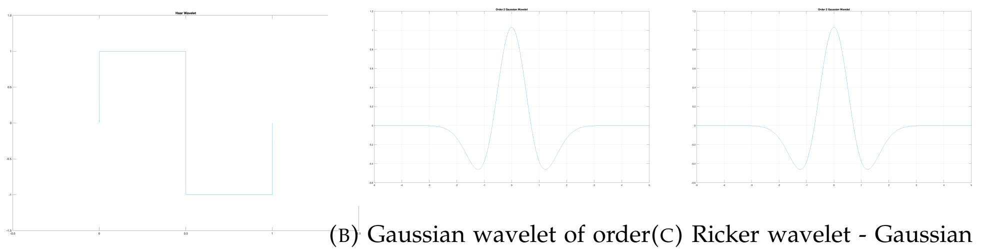
Wavelet transform analyzes a signal by its different frequency components at multiple resolution so features that are undiscovered at one resolution may be obvious at another. There are mainly 2 types of wavelet transforms, namely continuous wavelet transform (CWT) and discrete wavelet transform (DWT) and here are the mathematical representations for the two transforms: CWT finds how alike a wavelet is in a signal, given the above 2 properties that the wavelet has. **wavelet** This can be found by convolving the mother wavelet with our signal.

$$\mathrm{CWT}(a, b; x(t), \psi(t)) = \int_{-\infty}^{\infty} [x(t)\frac{1}{a} - \psi^*(\frac{t-b}{a})]dt$$

wavelet_denoise

where $x(t)$ is the original signal, $\psi(t)$ is the mother wavelet, a is a dilation parameter and b is a translation parameter. Dilation factor represents how dispersed the wavelet is (similar to scaling) while translation factor tells us where the wavelet is positioned in time (similar to shifting).

$$\begin{aligned} \mathrm{DWT}\ [n, a^j] &= \sum_{m=0}^{N-1} x[m].\psi_j{}^*[m-n], \\ \psi_j[n] &= \frac{1}{\sqrt{a^f}}\psi\left(\frac{n}{a^f}\right) \end{aligned} \tag{2}$$



(A) Haar wavelet (B) Gaussian wavelet of order 1 (C) Ricker wavelet - Gaussian wavelet of order 2

FIGURE 5.1: Examples of wavelets

where n is delay parameter, N is the length of signal, ψ is the discretized mother wavelet. **wavelet_denoise**

We will focus on DWT since computation is done on discrete wavelets so it requires less computational resources.

3. Spectral reduction

   Spectral noise gating learns from a noise profile and removes slow-changing tonal noise or hiss from the signal. In fact, this method is used by Audacity in its noise reduction algorithm. **audacity** Suppose noise is additive, and we can represent our noisy audio

$$(n) = x(n) + d(n), for 0 <= n <= N - 1 \tag{5.1}$$

where $x(n)$ is our original signal (signal we wish to recover), $d(n)$ is the noise, n is the discrete time index, N is the number of samples. Assuming $d(n)$ and $x(n)$ have no correlation, and we perform a short-time fourier transform on equation **CHANGE!!!1**:

$$Y(\omega, k) = X(\omega, k) + D(\omega, k)$$

where k is the frame number, which can be dropped if we assume the signal is segmented. Each segment will be of length N. We then have the desired signal in frequency domain:

$$X(\omega) = Y(\omega) - N(\omega)$$

Since the statistics of the noise is unknown, we try to find an estimate of noise spectrum by calculating the time-averaged noise spectrum using parts of the recording that only contain ambient noise. **reductionmanual**

$$N(\hat{\omega}) = \mathbf{E}[|N(\omega)|] = (1/N)\sum_{i=0}^{N-1} |N_i(\omega)|$$

We then get the estimated signal spectrum

$$X(\hat{\omega}) = Y(\omega) - N(\hat{\omega})$$

We then set a gain control for each frequency band so if the sound exceeded the threshold, the gain is set to 0 dB or a user-defined constant.

5.2.2 Choice of model and implementation

After trying to implement all 3 methods, a major difficulty encountered is that it is hard to set the parameters to implement for LPF and wavelet transform whilst spectral reduction works the best in removing the ambient noise. For example, since f_c depends on the pitch range of the user and the melody he/ she is inputting, finding an adequate f_c that separates desired frequencies from undesired ones is hard. As for wavelet transform, finding an adequate mother wavelet is a difficult task.
Although wavelet transform works better for real-life non-stationary signals compared to conventional frequency-based filters, if we do not feed a suitable mother wavelet, the performance is unsatisfactory, the model cannot distinguish between desired and undesired signals and will decrease P_signal at the same time, which is unfavourable when it comes to improving the SNR.
According to (D SHUKLA, 2003), there are 2 major concerns with using wavelet transform. Firstly, it is sensitive to shifting in time, even a minor shift will cause unpredictable change in transform coefficients which will then cause variations in the output signal. Secondly, wavelet transform suffers from poor directionality easily. For example, a 2-D DWT can only reveal 3 spatial-domain feature orientations, which limits the optimal representation of the signal.

Therefore, we decided to use spectral reduction as our noise filter algorithm. The implementation of spectral noise gating can be summarised according to (Karam et al., 2014) The noise spectrum $N(\omega)$ and its statistical measures are obtained by

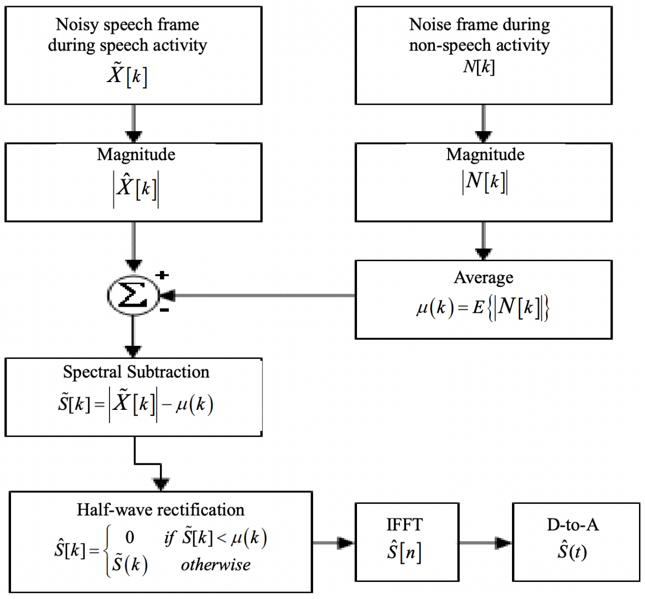

Noisy speech frame during speech activity $\tilde{X}[k]$
Noise frame during non-speech activity $N[k]$
Magnitude $|\hat{X}[k]|$
Magnitude $|N[k]|$
+
-
Average $\mu(k)=E\{|N[k]|\}$
Spectral Subtraction $\tilde{S}[k]=|\tilde{X}[k]|-\mu(k)$
Half-wave rectification $\hat{S}[k]=\begin{cases}0 & if\ \tilde{S}[k]<\mu(k)\\ \tilde{S}(k) & otherwise\end{cases}$
IFFT $\hat{S}[n]$
D-to-A $\hat{S}(t)$

FIGURE 5.2: Spectral noise gating flowchart **spectralflowchart**

asking users to record at least 3 seconds of silence before they sing into the app. Note that half-wave rectification is necessary after noise removal process of subtracting the average magnitude of noise spectrum. This is to target frequencies that have a higher average magnitude of noise spectrum $\mathbf{E}[|N(\omega)|]$ compared to that of the noisy speech spectrum $|X(\hat{\omega})|$. For those frequencies, we would replace the negative values with 0 with a half-wave rectification.

5.2.3 Improvements

A drawback with spectral reduction is that it does not handle extreme responses nicely. It does not reduce noises like squeaks. Also, since a half-wave rectification is included in the implementation process, (Rao & Sreelatha, 2014) has pointed out that residual noise will be created during the process of spectral reduction. Half-wave rectification introduces nonlinearity in the $X(\hat{\omega})$ spectrum and results in frequencies changing abruptly between frames.

As mentioned above, spectral reduction is built on the assumption that the noise is a stationary or slowly-varying. Yet in reality, there may be sudden squeaky noise in the background which is not recorded in the noise profile. In this case, spectral reduction cannot remove the squeak. To improve the situation, we will introduce a low-pass filter to filter out high-frequency noise. The reason for choosing LPF

	Male	Fe
Pitch range (Hz)	60-180	160
Praat pitch range (Hz)	50-300	100

TABLE 5.1: Praat pitch range

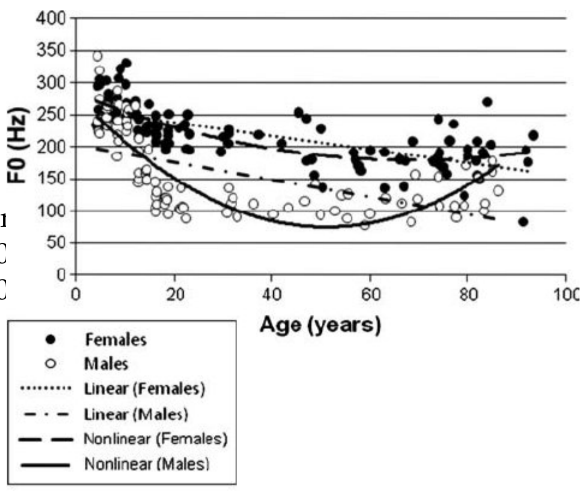


FIGURE 5.3: Scatter plot of fundamental frequency by age **f0age**

over a band-pass filter is that ambient noise is ususally low frequency and spectral reduction is effective in targeting the reduction of ambient noise. Thus as to avoid removing low-frequency desirable features, a low-pass filter will suffice.

To determine f_c for the LPF, it would be plausible to refer to biological features of the users, i.e. their age and gender. For males, the pitch level generally reduces from infancy to middle age, while a reversal of trend occurs after middle age. **womenprange** On the other hand, as pointed out by (Nishio Niimi, 2008), "Females in their 30s and 40s showed obviously lower frequencies than those in their 20s. Across all age groups, including the 80s, fundamental frequencies tended to decrease markedly in association with aging"

After referring to the measurements taken from 192 participants (Stathopoulos et al., 2011) and the pitch range set by Praat (a software for speech analysis) **praat**, a simple modelling of f_c according to age and gender is as below:

$$f_{c,male}(n) = 0.07n^2 - 7.5n + 280, for 4 <= n <= 93$$

$$f_{c,female}(n) = 0.02n^2 - 3n + 287, for 4 <= n <= 93$$

In deciding which filter to implement, there are 3 filters in consideration:

(a) Type 1 Chebyshev filter

$$G_n(\omega) = |H_n(j\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 T_n^2(\omega/\omega_0)}}$$

where ε is the ripple factor, ω_0 is the cut-off frequency and T_n is a Chebyshev polynomial of the nth order.

(b) Butterworth filter

$$G_n(\omega) = |H_n(j\omega)| = \frac{1}{sqrt1 + (\omega/\omega_0)^{2n})}$$

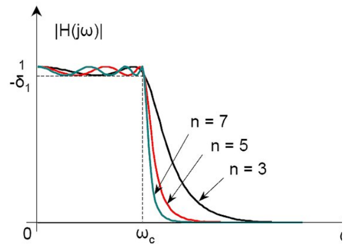
where ω_0 is the cut-.off frequency and n is the order of filter.

(c) Bessel filter

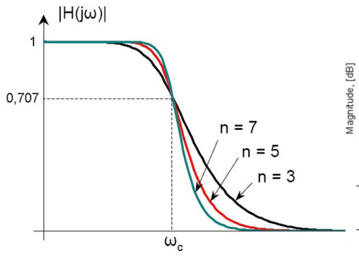
$$G_n(\omega) = |H_n(j\omega)| = \frac{\theta_n(0)}{\theta_n(j\omega/\omega_0)}$$

where $\theta_n(j\omega)$ is a reverse Bessel polynomial and ω_0 is the cut-off frequnecy

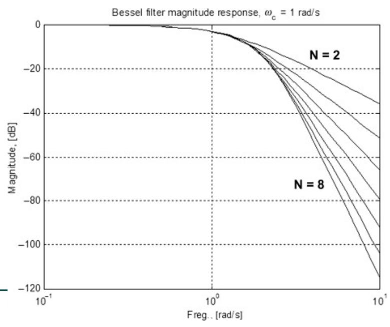
Chebyshev filter has a steeper roll-off compared to Butterworth and Bessel filter, but it also brings passband and stopband ripples, unlike Butterworth and Bessel filter which have a flat passband and stopband as they roll-offs towards zero. Moreover, Butterworth and Bessel filters have a better step response. Meanwhile, Bessel filter performs the best in step response since the overshoot is minimal. Also, an important characteristics Bessel filter has is that it introduces a linear-phase/ constant delay for $f < f_c$. This feature allows us to preserve the waveshape since all frequencies are delayed by the same amount. Thus, Bessel filter is preferred in this context.



(A) Type 1 Chebyshev filter frequency response



(B) Butterworth filter frequency response



(C) Bessel filter frequency response **bessel**

5.3 Pitch Detection Algorithm (PDA)

After removing noise, we pass the processed signal to a PDA to estimate the fundamental frequency ($f0$) of the signal.

5.3.1 Possible Models

There are 4 approaches to detect $f0$, which again can be classified into time domain and frequency domain.

1. Zero crossings (time domain) This is the most intuitive method to detecting pitch although it suffers from low accuracy. Assuming the input is monophonic, the fundamental frequency is estimated as:

   $$f0 = \frac{P_{zcr} f_s}{2N}$$

   where P_{Zcr} is the number of zero-crossing points, f_s is the sampling rate, N is the number of samples

2. YIN algorithm/ autocorrelation (time domain) As outlined by (de Cheveigné Kawahara, 2002), YIN algorithm is based on the autocorrelation method:

   $$r_t(\tau) = \sum_{j=t+1}^{t+W-\tau} x_j x_{j+\tau}$$

   where r$_t(\tau) istheautocorrelationfunctionoflag \tau calculatedattimeindext,Wistheintegrationwindows$

3. Harmonics (frequency domain)

4. CREPE (Convolutional Representation for Pitch Estimation)

5.3.2 Comparison between models and implementation

The zero crossings method will not be considered since it heavily relies on the assumption that the input audio is of pure tone. Yet,

"the human voice is not a pure tone (as produced by a tuning fork); rather, it is composed of a fundamental tone (or frequency of vibration) and a series of higher frequencies called upper harmonics," **humanmono**

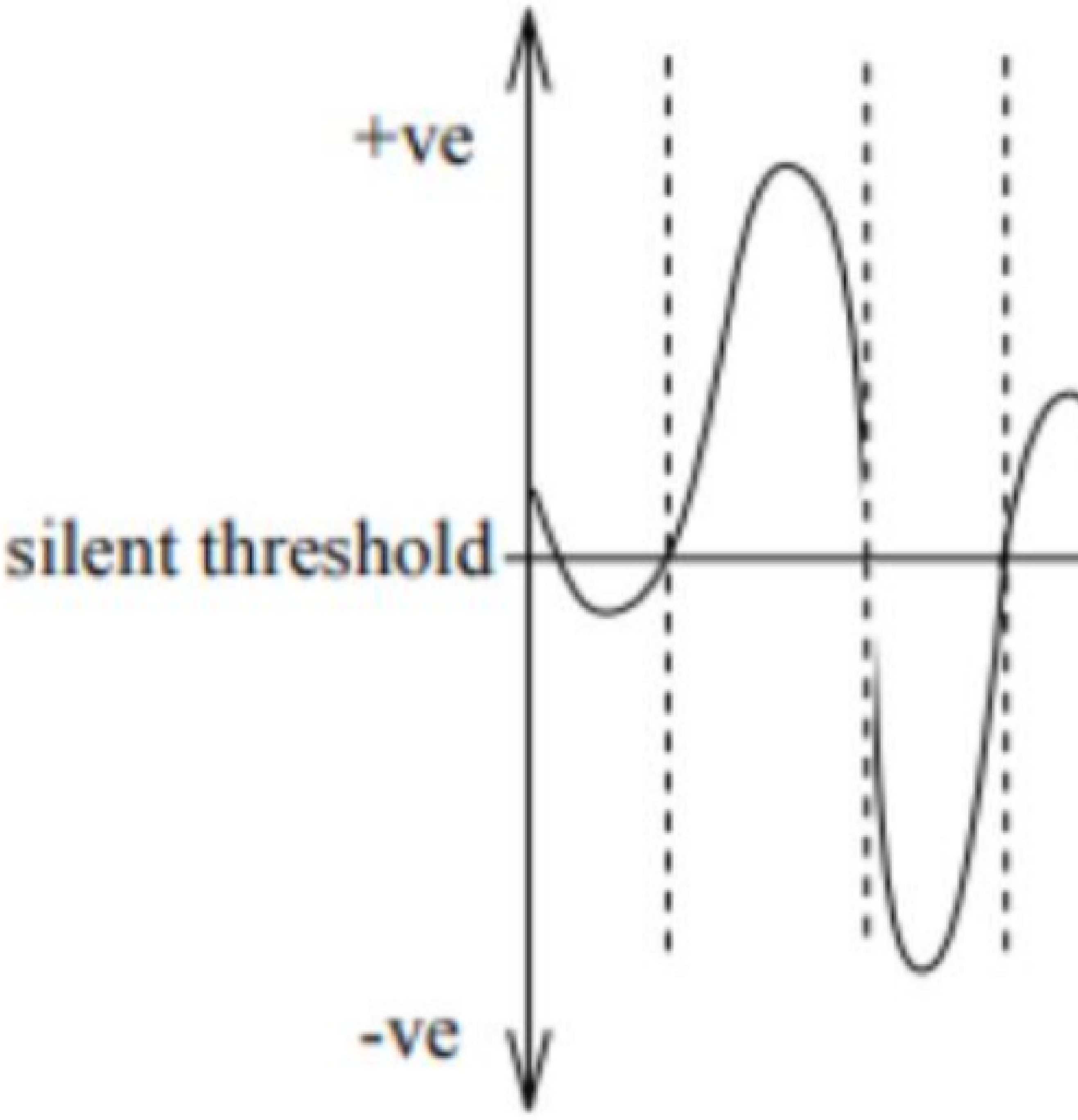


FIGURE 5.5: An example signal with zero-crossings marked in dotted lines **zcr**

5.3.3 Improvements

(find an algorithm to take in users' singing frequency to improve accuracy of the model (perhaps adjust the likelihood of the crepe model?) (look into automl)

5.4 Key Detection Algorithm (KDA)

5.4.1 Possible Models

5.4.2 Implementation

Chapter 6

Chapter Title Here

6.1 Welcome and Thank You

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

6.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

6.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on:
http://www.LaTeXTemplates.com
Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

6.2.2 A Short Math Guide for LaTeX

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here: http://www.ams.org/tex/amslatex.html under the "Additional Documentation" section towards the bottom of the page.

6.2.3 Common LaTeX Math Symbols

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page: http://www.sunilpatel.co.uk/latex-type/latex-math-symbols/

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

6.2.4 LaTeX on a Mac

The LaTeX distribution is available for many systems including Windows, Linux and Mac OS X. The package for OS X is called MacTeX and it contains all the applications you need – bundled together and pre-customized – for a fully working LaTeX environment and work flow.

MacTeX includes a custom dedicated LaTeX editor called TeXShop for writing your '`.tex`' files and BibDesk: a program to manage your references and create your bibliography section just as easily as managing songs and creating playlists in iTunes.

6.3 Getting Started with this Template

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the `main.tex` file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the `MastersDoctoralThesis.cls` file.

6.3.1 About this Template

This LaTeX Thesis Template is originally based and created around a LaTeX style file created by Steve R. Gunn from the University of Southampton (UK), department of Electronics and Computer Science. You can find his original thesis style file at his site, here: `http://www.ecs.soton.ac.uk/~srg/softwaretools/document/templates/`

Steve's `ecsthesis.cls` was then taken by Sunil Patel who modified it by creating a skeleton framework and folder structure to place the thesis files in. The resulting template can be found on Sunil's site here: `http://www.sunilpatel.co.uk/thesis-template`

Sunil's template was made available through `http://www.LaTeXTemplates.com` where it was modified many times based on user requests and questions. Version 2.0 and onwards of this template represents a major modification to Sunil's template and is, in fact, hardly recognisable. The work to make version 2.0 possible was carried out by Vel and Johannes Böttcher.

6.4 What this Template Includes

6.4.1 Folders

This template comes as a single zip file that expands out to several files and folders. The folder names are mostly self-explanatory:

Appendices – this is the folder where you put the appendices. Each appendix should go into its own separate `.tex` file. An example and template are included in the directory.

Chapters – this is the folder where you put the thesis chapters. A thesis usually has about six chapters, though there is no hard rule on this. Each chapter should go in its own separate `.tex` file and they can be split as:

- Chapter 1: Introduction to the thesis topic
- Chapter 2: Background information and theory
- Chapter 3: (Laboratory) experimental setup
- Chapter 4: Details of experiment 1
- Chapter 5: Details of experiment 2
- Chapter 6: Discussion of the experimental results
- Chapter 7: Conclusion and future directions

This chapter layout is specialised for the experimental sciences, your discipline may be different.

Figures – this folder contains all figures for the thesis. These are the final images that will go into the thesis document.

6.4.2 Files

Included are also several files, most of them are plain text and you can see their contents in a text editor. After initial compilation, you will see that more auxiliary files are created by LaTeX or BibTeX and which you don't need to delete or worry about:

example.bib – this is an important file that contains all the bibliographic information and references that you will be citing in the thesis for use with BibTeX. You can write it manually, but there are reference manager programs available that will create and manage it for you. Bibliographies in LaTeX are a large subject and you may need to read about BibTeX before starting with this. Many modern reference managers will allow you to export your references in BibTeX format which greatly eases the amount of work you have to do.

MastersDoctoralThesis.cls – this is an important file. It is the class file that tells LaTeX how to format the thesis.

main.pdf – this is your beautifully typeset thesis (in the PDF file format) created by LaTeX. It is supplied in the PDF with the template and after you compile the template you should get an identical version.

main.tex – this is an important file. This is the file that you tell LaTeX to compile to produce your thesis as a PDF file. It contains the framework and constructs that tell LaTeX how to layout the thesis. It is heavily commented so you can read exactly what each line of code does and why it is there. After you put your own information into the *THESIS INFORMATION* block – you have now started your thesis!

Files that are *not* included, but are created by LaTeX as auxiliary files include:

main.aux – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file.

main.bbl – this is an auxiliary file generated by BibTeX, if it is deleted, BibTeX simply regenerates it when you run the `main.aux` file. Whereas the `.bib` file contains all the references you have, this `.bbl` file contains the references you have actually cited in the thesis and is used to build the bibliography section of the thesis.

main.blg – this is an auxiliary file generated by BibTeX, if it is deleted BibTeX simply regenerates it when you run the main `.aux` file.

main.lof – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It tells LaTeX how to build the *List of Figures* section.

main.log – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It contains messages from LaTeX, if you receive errors and warnings from LaTeX, they will be in this `.log` file.

main.lot – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It tells LaTeX how to build the *List of Tables* section.

main.out – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file.

So from this long list, only the files with the `.bib`, `.cls` and `.tex` extensions are the most important ones. The other auxiliary files can be ignored or deleted as LaTeX and BibTeX will regenerate them.

6.5 Filling in Your Information in the `main.tex` File

You will need to personalise the thesis template and make it your own by filling in your own information. This is done by editing the `main.tex` file in a text editor or your favourite LaTeX environment.

Open the file and scroll down to the third large block titled *THESIS INFORMATION* where you can see the entries for *University Name*, *Department Name*, etc …

Fill out the information about yourself, your group and institution. You can also insert web links, if you do, make sure you use the full URL, including the `http://` for this. If you don't want these to be linked, simply remove the `\href{url}{name}` and only leave the name.

When you have done this, save the file and recompile `main.tex`. All the information you filled in should now be in the PDF, complete with web links. You can now begin your thesis proper!

6.6 The `main.tex` File Explained

The `main.tex` file contains the structure of the thesis. There are plenty of written comments that explain what pages, sections and formatting the LaTeX code is creating. Each major document element is divided into commented blocks with titles in all capitals to make it obvious what the following bit of code is doing. Initially there seems to be a lot of LaTeX code, but this is all formatting, and it has all been taken care of so you don't have to do it.

Begin by checking that your information on the title page is correct. For the thesis declaration, your institution may insist on something different than the text given. If this is the case, just replace what you see with what is required in the *DECLARATION PAGE* block.

Then comes a page which contains a funny quote. You can put your own, or quote your favourite scientist, author, person, and so on. Make sure to put the name of the person who you took the quote from.

Following this is the abstract page which summarises your work in a condensed way and can almost be used as a standalone document to describe what you have done. The text you write will cause the heading to move up so don't worry about running out of space.

Next come the acknowledgements. On this page, write about all the people who you wish to thank (not forgetting parents, partners and your advisor/supervisor).

The contents pages, list of figures and tables are all taken care of for you and do not need to be manually created or edited. The next set of pages are more likely to be optional and can be deleted since they are for a more technical thesis: insert a list of abbreviations you have used in the thesis, then a list of the physical constants and numbers you refer to and finally, a list of mathematical symbols used in any formulae. Making the effort to fill these tables means the reader has a one-stop place to refer to instead of searching the internet and references to try and find out what you meant by certain abbreviations or symbols.

The list of symbols is split into the Roman and Greek alphabets. Whereas the abbreviations and symbols ought to be listed in alphabetical order (and this is *not* done automatically for you) the list of physical constants should be grouped into similar themes.

The next page contains a one line dedication. Who will you dedicate your thesis to?

Finally, there is the block where the chapters are included. Uncomment the lines (delete the `%` character) as you write the chapters. Each chapter should be written in its own file and put into the *Chapters* folder and named `Chapter1`, `Chapter2`, etc...Similarly for the appendices, uncomment the lines as you need them. Each appendix should go into its own file and placed in the *Appendices* folder.

After the preamble, chapters and appendices finally comes the bibliography. The bibliography style (called *`authoryear`*) is used for the bibliography and is a fully featured style that will even include links to where the referenced paper can be found online. Do not underestimate how grateful your reader will be to find that a reference to a paper is just a click away. Of course, this relies on you putting the URL information into the BibTeX file in the first place.

6.7 Thesis Features and Conventions

To get the best out of this template, there are a few conventions that you may want to follow.

One of the most important (and most difficult) things to keep track of in such a long document as a thesis is consistency. Using certain conventions and ways of doing things (such as using a Todo list) makes the job easier. Of course, all of these are optional and you can adopt your own method.

6.7.1 Printing Format

This thesis template is designed for double sided printing (i.e. content on the front and back of pages) as most theses are printed and bound this way. Switching to one sided printing is as simple as uncommenting the *`oneside`* option of the `documentclass` command at the top of the `main.tex` file. You may then wish to adjust the margins to suit specifications from your institution.

The headers for the pages contain the page number on the outer side (so it is easy to flick through to the page you want) and the chapter name on the inner side.

The text is set to 11 point by default with single line spacing, again, you can tune the text size and spacing should you want or need to using the options at the very start of `main.tex`. The spacing can be changed similarly by replacing the *`singlespacing`* with *`onehalfspacing`* or *`doublespacing`*.

6.7.2 Using US Letter Paper

The paper size used in the template is A4, which is the standard size in Europe. If you are using this thesis template elsewhere and particularly in the United States, then you may have to change the A4 paper size to the US Letter size. This can be done in the margins settings section in `main.tex`.

Due to the differences in the paper size, the resulting margins may be different to what you like or require (as it is common for institutions to dictate certain margin sizes). If this is the case, then the margin sizes can be tweaked by modifying the values in the same block as where you set the paper size. Now your document should be set up for US Letter paper size with suitable margins.

6.7.3 References

The `biblatex` package is used to format the bibliography and inserts references such as this one (**Reference1**). The options used in the `main.tex` file mean that the in-text citations of references are formatted with the author(s) listed with the date of the publication. Multiple references are separated by semicolons (e.g. (**Reference2**; **Reference1**)) and references with more than three authors only show the first author with *et al.* indicating there are more authors (e.g. (**Reference3**)). This is done automatically for you. To see how you use references, have a look at the `Chapter1.tex` source file. Many reference managers allow you to simply drag the reference into the document as you type.

Scientific references should come *before* the punctuation mark if there is one (such as a comma or period). The same goes for footnotes[1]. You can change this but the most important thing is to keep the convention consistent throughout the thesis. Footnotes themselves should be full, descriptive sentences (beginning with a capital letter and ending with a full stop). The APA6 states: "Footnote numbers should be superscripted, [...], following any punctuation mark except a dash." The Chicago manual of style states: "A note number should be placed at the end of a sentence or clause. The number follows any punctuation mark except the dash, which it precedes. It follows a closing parenthesis."

The bibliography is typeset with references listed in alphabetical order by the first author's last name. This is similar to the APA referencing style. To see how LaTeX typesets the bibliography, have a look at the very end of this document (or just click on the reference number links in in-text citations).

A Note on bibtex

The bibtex backend used in the template by default does not correctly handle unicode character encoding (i.e. "international" characters). You may see a warning about this in the compilation log and, if your references contain unicode characters, they may not show up correctly or at all. The solution to this is to use the biber backend instead of the outdated bibtex backend. This is done by finding this in `main.tex`: *`backend=bibtex`* and changing it to *`backend=biber`*. You will then need to delete all auxiliary BibTeX files and navigate to the template directory in your terminal (command prompt). Once there, simply type `biber main` and biber will compile your bibliography. You can then compile `main.tex` as normal and your bibliography will be updated. An alternative is to set up your LaTeX editor to compile with biber instead of bibtex, see here for how to do this for various editors.

6.7.4 Tables

Tables are an important way of displaying your results, below is an example table which was generated with this code:

[1]Such as this footnote, here down at the bottom of the page.

TABLE 6.1: The effects of treatments X and Y on the four groups studied.

Groups	**Treatment X**	**Treatment Y**
1	0.2	0.8
2	0.17	0.7
3	0.24	0.75
4	0.68	0.3

```
\begin{table}
\caption{The effects of treatments X and Y on the four groups studied.}
\label{tab:treatments}
\centering
\begin{tabular}{l l l}
\toprule
\tabhead{Groups} & \tabhead{Treatment X} & \tabhead{Treatment Y} \\
\midrule
1 & 0.2 & 0.8\\
2 & 0.17 & 0.7\\
3 & 0.24 & 0.75\\
4 & 0.68 & 0.3\\
\bottomrule\\
\end{tabular}
\end{table}
```

You can reference tables with `\ref{<label>}` where the label is defined within the table environment. See `Chapter1.tex` for an example of the label and citation (e.g. Table **??**).

6.7.5 Figures

There will hopefully be many figures in your thesis (that should be placed in the *Figures* folder). The way to insert figures into your thesis is to use a code template like this:

```
\begin{figure}
\centering
\includegraphics{Figures/Electron}
```

```
\decoRule
\caption[An Electron]{An electron (artist's impression).}
\label{fig:Electron}
\end{figure}
```

Also look in the source file. Putting this code into the source file produces the picture of the electron that you can see in the figure below.

Sometimes figures don't always appear where you write them in the source. The placement depends on how much space there is on the page for the figure. Sometimes there is not enough room to fit a figure directly where it should go (in relation to the text) and so LaTeX puts it at the top of the next page. Positioning figures is the job of LaTeX and so you should only worry about making them look good!

Figures usually should have captions just in case you need to refer to them (such as in Figure **??**). The `\caption` command contains two parts, the first part, inside the square brackets is the title that will appear in the *List of Figures*, and so should be short. The second part in the curly brackets should contain the longer and more descriptive caption text.

The `\decoRule` command is optional and simply puts an aesthetic horizontal line below the image. If you do this for one image, do it for all of them.

LaTeX is capable of using images in pdf, jpg and png format.

6.7.6 Typesetting mathematics

If your thesis is going to contain heavy mathematical content, be sure that LaTeX will make it look beautiful, even though it won't be able to solve the equations for you.

The "Not So Short Introduction to LaTeX" (available on CTAN) should tell you everything you need to know for most cases of typesetting mathematics. If you need more information, a much more thorough mathematical guide is available from the AMS called, "A Short Math Guide to LaTeX" and can be downloaded from: `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`

There are many different LaTeX symbols to remember, luckily you can find the most common symbols in The Comprehensive LaTeX Symbol List.

You can write an equation, which is automatically given an equation number by LaTeX like this:

```
\begin{equation}
E = mc^{2}
\label{eqn:Einstein}
\end{equation}
```

This will produce Einstein's famous energy-matter equivalence equation:

$$E = mc^2 \tag{6.1}$$

All equations you write (which are not in the middle of paragraph text) are automatically given equation numbers by LaTeX. If you don't want a particular equation numbered, use the unnumbered form:

```
\[ a^{2}=4 \]
```

6.8 Sectioning and Subsectioning

You should break your thesis up into nice, bite-sized sections and subsections. LaTeX automatically builds a table of Contents by looking at all the `\chapter{}`, `\section{}` and `\subsection{}` commands you write in the source.

The Table of Contents should only list the sections to three (3) levels. A `chapter{}` is level zero (0). A `\section{}` is level one (1) and so a `\subsection{}` is level two (2). In your thesis it is likely that you will even use a `subsubsection{}`, which is level three (3). The depth to which the Table of Contents is formatted is set within `MastersDoctoralThesis.cls`. If you need this changed, you can do it in `main.tex`.

6.9 In Closing

You have reached the end of this mini-guide. You can now rename or overwrite this pdf file and begin writing your own `Chapter1.tex` and the rest of your thesis. The easy work of setting up the structure and framework has been taken care of for you. It's now your job to fill it out!

Good luck and have lots of fun!

Guide written by —

Sunil Patel: www.sunilpatel.co.uk

Vel: LaTeXTemplates.com

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or

`\hypersetup{citecolor=green}`, or

`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:

`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.