

The final report for a dissertation that will be submitted in partial fulfilment of a  
University of Greenwich Master's Degree

**An investigation into the generative composition of music by  
means of a LSTM Recurrent Neural Network based  
algorithm.**

**Name: Joshua Roses-Agoro**

**Student ID: 000992718**

**Programme of Study: MSc Data Science**

## **Final Report**

**Date Proposal Submitted: 01/06/2021**

**Project Hand In Date: 23/09/2021**

**Supervisor: Dr Tuan Vuong**

# Acknowledgements

Firstly, I would like to acknowledge Dr Tuan Vuong who has been my supervisor for the duration of my master's degree. An immense amount of gratitude is dedicated toward the mentorship that you have provided me with throughout the previous year. Your guidance and support have pushed me to explore new avenues in the world of data science and as a result significantly increased my knowledge and understanding.

I would not have been able to produce this piece of work without your constant feedback and encouragement that inspired me to achieve more.

I would also like to thank my friend and colleague Michael Briggs who has been with me on this journey since the beginning of our bachelor's degrees. Your persistence of pushing me in the right direction and competitive nature always inspired me to want more out of my education. And our ability to share ideas and help one another to formulate better work has been a blessing and something I never took for granted.

Throughout this year and the many challenges, it has presented, I wouldn't have been able to complete my studies without the support of my family who always took pride in my achievements and saw the best in me when I wasn't able to, it is because of their love and encouragement that I had the strength to continue until the very end.

My final thanks and deepest indebtedness belongs to my loving partner who has been there along every step of the journey from start to end of my master's degree. It was your belief in me that gave me the courage to take the initial steps that began this postgraduate journey, and throughout the ups and downs of our relationship your love and belief in me has never ceased, and it is due to this reason that I dedicate my magnum opus to you.

# Abstract

The implementation of Artificial Intelligence in the domain of music has been a controversial topic in the eyes of many members of the music industry, but recent developments have shown the robust capabilities of Machine Learning in this field, which have led to improvements in the way algorithms are written to tackle this scenario.

In this thesis, the usage of a Long Short-Term Memory based algorithm has been explored, with the intention of generative music composition by means of a Recurrent Neural Network. The proposed approach makes use of the Python based musicology toolkit Music21, to extract the tonal properties of a MIDI file-based dataset that contains 80 samples taken from the Pokémon games soundtrack, as this music contains distinct melodies which are typically written on the same scale, thus maintaining a structure.

The categorical data from the dataset is mapped to integer-based numerical data, followed by input and output sequences being prepared for the neural network. The way in which this works is by calculating a formula that determines which note should be output and when, and this data is fed to the network.

Once the model is trained, the generation of notes occurs by first reversing the previously implemented mapping function, thus restoring the data to its string-based categorical format. Followed by neural network inference in which notes will be generated from the input/output sequences created before. The generated notes are decoded and parsed through Music21 to form a distinction between notes and chords, and the objects are written to a MIDI file to produce the final output.

Evaluating the generated music piece is faced with two potential approaches, summative/subjective and formative/objective. Due to the characteristics of a musical piece which is built around creativity, the path chosen was summative with a user study created around a listening based experiment, conducted on 10 participants with musical backgrounds to evaluate the audio piece based on several criteria. The results yielded above average satisfactory scores with the overall impression being positive.

# List of Figures

|   |    |
|---|----|
| Figure 1 – Full Piano Keyboard.....   | 17 |
| Figure 2 - Proposed workflow of the formative approach to evaluation - (Yang, 2018) .....                   | 30 |
| Figure 3 - Pipeline Workflow .....  | 43 |
| Figure 4 - Pipeline Architecture/Process .....  | 44 |
| Figure 5 - Excerpt from a MIDI file within the dataset .....  | 46 |
| Figure 6 – Heptatonic versus Pentatonic .....   | 47 |
| Figure 7 - Separating the note/chord objects in each MIDI file .....  | 48 |
| Figure 8 - Mapping String-based Categorical to Integer-based Numerical .....                                | 50 |
| Figure 9 - Mapping Function .....   | 50 |
| Figure 10 - Preparation of the input and output sequences used by the network .....                         | 51 |
| Figure 11 - One-hot Encoding .....  | 52 |
| Figure 12 - LSTM Network Architecture - (Zaytar, 2016).....   | 53 |
| Figure 13 - Architecture of the network used for training .....   | 55 |
| Figure 14 - Loss = Categorical Cross Entropy .....  | 57 |
| Figure 15 - RMSprop Update Rule .....   | 57 |
| Figure 16 - Training the Model.....   | 58 |
| Figure 17 - Mapping Integer-based Numerical to String-based Categorical.....                                | 60 |
| Figure 18 - Generating notes from the training set .....  | 61 |
| Figure 19 - Note Generation from the Input Sequences .....  | 62 |
| Figure 20 - E.g. How notes are chosen based on frequency of occurrence .....                                | 63 |
| Figure 21 - Looping through the generated sequences to separate note/chord objects into stream objects..... | 64 |
| Figure 22 - Notes/Chords within stream object are written to an output in MIDI format .....                 | 65 |
| Figure 23 - Sheet music of file 'project_output_3.MID' .....  | 69 |
| Figure 24 - Identification of Middle C on the full piano keyboard .....                                     | 69 |
| Figure 25 - Results of the user-based study .....   | 71 |
| Figure 26 - Mean value of the results obtained from the user study.....                                     | 71 |
| Figure 27 - Dataset Sample Line Plot .....  | 77 |
| Figure 28 - Dataset Sample Histogram .....  | 77 |
| Figure 29 - Dataset Sample Scatter Plot .....   | 78 |

|  |     |
|--|-----|
| Figure 30 - Dataset Sample Composition Parameters .....  | 78  |
| Figure 31 - Generated Output Line Plot .....             | 79  |
| Figure 32 - Generated Output Histogram .....             | 79  |
| Figure 33 - Generated Output Composition Parameters..... | 80  |
| Figure 34 - Generated Output Scatter Plot.....           | 80  |
| Figure 35 - File structure of the Implementation .....   | 100 |

## List of Tables

|  |    |
|--|----|
| Table 1 - Time Constraints .....                       | 38 |
| Table 2 - Project Timeline .....                       | 38 |
| Table 3 - Functional Requirements.....                 | 40 |
| Table 4 - Non-functional Requirements .....            | 40 |
| Table 5 - Packages Required.....                       | 42 |
| Table 6 - Hardware Specifications.....                 | 43 |
| Table 7 - Evaluation of the Algorithm .....            | 66 |
| Table 8 - Verbal Participant Feedback (Optional) ..... | 74 |
| Table 9 - Functional Requirement Fulfilment .....      | 83 |
| Table 10 - Non-functional Requirement Fulfilment ..... | 83 |
| Table 11 - Results from Training the Network.....      | 86 |

# List of Acronyms

## Cases

|  |    |
|--|----|
| AI : Artificial Intelligence.....                | 11 |
| ANNs : Artifical Neural Networks .....           | 23 |
| CUDA : Compute Unified Device Architecture ..... | 39 |
| DSDM : Dynamic Systems Development Method.....   | 19 |
| EMI : Experiments in Musical Intelligence .....  | 21 |
| FNNs : Feed-Forward Neural Networks.....         | 23 |
| GPU : Graphics Processing Unit.....              | 40 |
| GUI : Graphical User Interface.....              | 22 |
| IPR : Intellectual Property Rights .....         | 33 |
| LSTM : Long Short-Term Memory .....              | 9  |
| MIDI : Musical Intrument Digital Interface ..... | 9  |
| MoSCoW : Must Have, Should Have, Could .....     | 10 |
| Relu : Rectified Linear Activation Function..... | 54 |
| RMSprop : Root Mean Squared Propagation .....    | 55 |
| RNNs : Recurrent Neural Networks .....           | 23 |
| SNNs : Simulated Neural Networks .....           | 23 |

# Contents

## **I. Acknowledgements**

## **II. Abstract**

## **III. List-of-Figures**

## **IV. List-of-Tables**

## **V. List-of-Acronyms**

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction .....</b>                                 | <b>11</b> |
| 1.1      | Background.....   | 13        |
| 1.2      | Scope of this Project.....                                | 15        |
| 1.3      | Music Background Information .....                        | 16        |
| 1.4      | Research Questions and Objectives .....                   | 19        |
| 1.5      | Approach to Project Delivery .....                        | 21        |
| <b>2</b> | <b>Literature Review .....</b>                            | <b>22</b> |
| 2.1      | Approach to Literature Searching .....                    | 22        |
| 2.2      | Identifying the Problem.....                              | 22        |
| 2.3      | Artificial Intelligence in the domain of Music.....       | 23        |
| 2.4      | ANN's and their Perspective Architectures .....           | 25        |
| 2.4.1    | Neural Networks.....                                      | 25        |
| 2.4.2    | Recurrent NN's .....                                      | 26        |
| 2.4.3    | LSTM NN's.....  | 27        |
| 2.5      | Methods for Evaluation on Generative Models in Music..... | 28        |
| 2.5.1    | Approach .....  | 29        |
| 2.5.2    | Summative.....  | 30        |
| 2.5.3    | Formative.....  | 31        |
| 2.6      | Review of Related Work .....                              | 32        |
| 2.7      | Conclusions.....  | 34        |



|       |   |    |
|-------|---|----|
| 3     | <b>Legal, Social and Ethical Issues</b>   | 35 |
| 4     | <b>Design</b>                             | 37 |
| 4.1   | Introduction                              | 37 |
| 4.2   | Initial Planning                          | 37 |
| 4.3   | Analysis of Requirements                  | 39 |
| 4.3.1 | Functional Requirements                   | 40 |
| 4.3.2 | Non-functional Requirements               | 40 |
| 4.4   | Overview of System Architecture           | 41 |
| 4.4.1 | Software                                  | 41 |
| 4.4.2 | Hardware                                  | 42 |
| 4.4.3 | Proposed Model Architecture               | 43 |
| 4.4.4 | Proposed Model Workflow                   | 44 |
| 5     | <b>Implementation</b>                     | 46 |
| 5.1   | Introduction                              | 46 |
| 5.2   | Training Data                             | 46 |
| 5.3   | Dataset Preparation                       | 48 |
| 5.4   | Pipeline Development & Feature Extraction | 49 |
| 5.4.1 | Mapping Function                          | 49 |
| 5.4.2 | Input/Output Sequences for the Network    | 51 |
| 5.5   | Network Architecture & Training           | 53 |
| 5.5.1 | Network Layers                            | 53 |
| 5.5.2 | Network Specification                     | 54 |
| 5.5.3 | Defining Model Parameters                 | 55 |
| 5.5.4 | Fitting the Model                         | 58 |
| 5.6   | Prediction & Generating the Final Output  | 60 |
| 5.6.1 | Reverse Mapping Function                  | 60 |
| 5.6.2 | Generating Notes                          | 61 |
| 5.6.3 | Write Notes to MIDI file                  | 64 |
| 6     | <b>Evaluation &amp; Results</b>           | 66 |
| 6.1   | Approach to Evaluation of the Algorithm   | 66 |
| 6.2   | Approach to Subjective Evaluation         | 68 |
| 6.2.1 | Listening Based Experiment                | 68 |

|       |  |     |
|-------|--|-----|
| 6.2.2 | Sheet Music of the excerpt that was Evaluated .....                        | 69  |
| 6.2.3 | User-based Study Criteria & Results.....                                   | 70  |
| 6.2.4 | Mean Value of Results .....  | 71  |
| 6.2.5 | Participant Feedback .....   | 73  |
| 6.3   | Approach to Objective Evaluation.....                                      | 75  |
| 6.3.1 | Objective Evaluation of a Sample from the Dataset .....                    | 77  |
| 6.3.2 | Objective Evaluation of a Sample from the Generated Output .....           | 79  |
| 6.3.3 | Conclusions on Objective Evaluation .....                                  | 81  |
| 7     | <b>Conclusion</b> .....  | 82  |
| 7.1   | Overview.....  | 82  |
| 7.2   | Requirement Fulfilment .....   | 83  |
| 7.3   | Findings and Recommendations .....   | 85  |
| 7.4   | Neglected Considerations & Further Development .....                       | 90  |
| 7.4.1 | Increase the size of the Dataset .....                                     | 90  |
| 7.4.2 | Adding classes to handle varying note lengths/durations .....              | 90  |
| 7.4.3 | Establishing a beginning and ending to each generated piece .....          | 91  |
| 7.4.4 | More in-depth objective evaluation methods .....                           | 91  |
| 7.5   | Evaluation of Personal Performance.....                                    | 92  |
| 8     | <b>References</b> .....  | 94  |
| 9     | <b>Bibliography</b> .....  | 96  |
| 10    | <b>Appendices</b> .....  | 100 |
| 10.1  | Appendix A – Musical Excerpt from the Listening-Based Experiment .....     | 100 |
| 10.2  | Appendix B – Alternate MIDI output that was generated by the Network ..... | 100 |
| 10.3  | Appendix C – Sample from the Dataset .....                                 | 100 |
| 10.4  | Appendix D – File structure of the Implementation.....                     | 100 |

# 1 Introduction

Does artificial intelligence possess the capability of creativity when enhanced by machine learning? This is a controversial question with many valid arguments on both ends of the spectrum. This question will be addressed during this thesis, and the final outcome of the implementation will either prove or disprove that hypothesis.

The primary focus of this thesis is an investigation into the composition of music by means of a LSTM Recurrent Neural Network based algorithm, followed by a composite implementation, consisting of two phases, training, and prediction.

“Recurrent neural networks, of which LSTMs (“long short-term memory” units) are the most powerful and well-known subset, are a type of artificial neural network designed to recognize patterns in sequences of data, such as numerical times series data” - (Nicholson, 2020)

LSTMs are one of the most powerful and useful neural networks, with the ability to demonstrate exceptional dexterity on numerous use cases, particularly on the subject of sequential data, and it is this affinity that makes this branch of neural networks so appropriate for the problem at hand. Which is interpreting a mid-sized dataset consisting of almost 100 MIDI files, translating the notes present into binary format, defining a data dictionary that prepares the sequences of data present in the dataset for the network, and finally feeding the data into the network. A function then chooses a random place within the sequence of data and produces an output which generalises an entirely new sound based on the train set.

The purpose of this document is to provide a concise and succinct argument which reinforces the claim made in the initial prognosis. The way in which this will be achieved is by means of thorough investigation that can be divided into several sections, the intention behind this is to increase the structural integrity of the arguments being presented. One of these sections will be a literature review, in which pre-existing work on the subject matter is reviewed, surveyed, and critically analyzed with the objective of formulating an intelligent argument capable of reinforcing the thesis itself.

This will be followed by a section on pre-existing products which are similar in design and purpose to that which has been developed throughout the duration of this project. Comparing and contrasting the most sophisticated software available against what has been produced in this project is a means of gauging a potential gap in the market and any ineptitudes with pre-existing software, and what has been developed for the project.

Next, a section on the technologies that were used to achieve the implementation in this project, and a justification which reinforces why they were chosen. In reviewing the other available technologies, a contrast will also be made against that which was chosen for the project, with the aim of reinforcing the decisions that were made.

A thorough requirement analysis will be in the next section, followed by a requirement specification, which outlines the particular functionalities required to make this implementation a success. The requirement specification will also consist of MoSCoW prioritization, informing the reader of the priority of the aforementioned requirements. This section also fulfils part of the design documentation because it defines the specific parameters that are essential when it comes to the developmental stage of the project.

Finally, the document will conclude with a thorough analysis and evaluation, with the intention of identifying the ways in which the project was a success, and how it could be improved. As well as the contributing factors which aided in the success.

## 1.1 Background

Artificial Intelligence has generally been considered to lack the capability of creativity, and has always required some level of human input in that regard for creative directive. But deeper investigation into what truly defines creativity has pushed researchers to pioneer new methodologies and technologies capable of more than just replication when it comes to music.

“This perspective is particularly characteristic of authors who embrace the mathematical style of harmonic theory, in which musical intervals are expressed as ratios of numbers, and who hold that the principles to which musically acceptable structures and relations must conform are principles proper to mathematics.” - (Barker, 2010)

The views of Andrew Barker similarly align with both the philosophers Aristotle and Boethius, who argued that music are numbers made audible, and this statement holds true value when we consider that the foundation of music itself fundamentally comprises of symbols, structures and processes. Thus, identifying an avenue where AI is able to show its capabilities through the enhancement of machine learning in which the structures and processes of music can be interpreted and understood by algorithms that can break it down into digestible sequences.

The emotional aspect of music is what leads many to say that without human intervention a computer would not be able to replicate such, but there is no denying the major rational component that is the very foundation of music, and machine learning is an ideal vessel to realise the exploration of the genuine capabilities of artificial intelligence on the subject matter of music.

The research conducted in this thesis is essential in both the progression of music and artificial intelligence, because both fields are continuously evolving, pushing both humans and technology to new heights. And although many studies have been done on the capability of artificial intelligence regarding its ability to develop music when reinforced by machine learning, none of these studies have addressed its applicability and usefulness in a real-world scenario, and then gone even further in the sense of

building upon the achieved outcomes as to improve the development and application of machine learning. This thesis aims to address that exact gap which is missing in current research by analysing the results of the output and evaluating it by addressing how useful the algorithm could be in a real-world setting.

The instrument of choice when it came to this thesis was the piano because of the simplistic nature of its notes and chords which appear in very apparent and distinct sequences. Basing the algorithm on an instrument with a very firm foundation in sequences and structure is absolutely essential in ensuring the accuracy of the algorithm's output, because it predicts based on a set amount of previously interpreted notes.

I personally have an extreme passion for music and it has been an essential part of my upbringing. As a child I received guitar lessons, piano lessons, and drum lessons. And I went so far as to complete my grade 8 examinations for drumming as this has always been my most proficient instrument. I have the knowledge and expertise on music and its theory to carry out the implementation of this thesis in a decisive manner.

## 1.2 Scope of this Project

The intention of this project is to successfully develop and fully implement a deep neural network, more specifically, a Long Short-Term Memory (LSTM) algorithm which is capable of producing music of a particular style by training it according to music of that genre. The ultimate objective being to determine the applicability of Artificial Intelligence (AI) within the world of music, and to test the bounds and capabilities of the LSTM algorithm and how well it is able to generalize according to a specific amount of training.

For my bachelor's degree dissertation, I originally had the idea of music composition by utilization of Machine Learning and although my supervisor encouraged me to explore this idea, I eventually made the decision that it could potentially be too difficult to achieve in the period of time that had been assigned and with the skills I had at that time. At the present date with more knowledge and skills than I had before, I have decided to tackle this topic for my master's dissertation, with the approval of Dr Vuong.

The topic of AI in the world of music is somewhat controversial in the eyes of some members of the music industry, with the concern of AI becoming so advanced that it invalidates their jobs. But forward thinkers believe that integrating human capability with that of AI is the most optimal way forward to achieve new feats.

The primary research for this project will be conducted on the most optimal and efficient way to develop and implement the algorithm itself, which is the core focus behind generating an algorithm which is capable of being trained to generalize and create its own unique musical output based on the specific bounds of the training set. The next avenue of research will be a comparison of the applicable technologies capable of producing an appropriate outcome for this project and determining which is best suited to the completion of the task based on the relative strengths and weaknesses of each methodology. And the final avenue of research will be based on the applicability of AI in the music industry, as was mentioned in the opening paragraph of this section. And what pros and cons this may lead to in both a societal sense and in a technological sense.

The build itself will be implemented on my own machine locally, as I have the hardware capable for developing and training the algorithm. The algorithm will be designed using Python 3.7, a programming language, with TensorFlow 2.2.1, a Machine Learning library, and Keras, which is an open-source software library that provides a Python interface for artificial neural networks. The deep neural network will make use of the LSTM methodology which is based on an artificial Recurrent Neural Network architecture. The reason this specific methodology will be used is because it has the ability to process entire sequences of data, which is exactly what music is, instead of singular data points.

### **1.3 Music Background Information**

In order to comprehend and genuinely understand some of the key features of this project, there is a requirement for at the very least a basic understanding of some of the key concepts in relation to music theory. The highness or lowness of a sound can be described as pitch, the sound being referred to is made up of notes, and in music there are 7 distinct notes which are A, B, C, D, E, F, G.

A set of notes ranging from A to G belongs to an octave, and an octave is used to designate which pitch range any given particular notes may be in. There are 88 playable keys which exist within a piano, and these 88 keys exist within 7 octaves, spanning the entire keyboard.

When a group of notes are played simultaneously as a group, this action can be referred to as a chord and it produces a distinct sound. Music pieces typically exist in a specific key, which can be defined as the note that forms the basis for the music within the piece. And lastly, a scale is a specific set of notes built on the base of the key, which particularly occur as an ordered sequence.

As part of this understanding, a few critical music related terminologies must be understood in depth. And they are as follows:



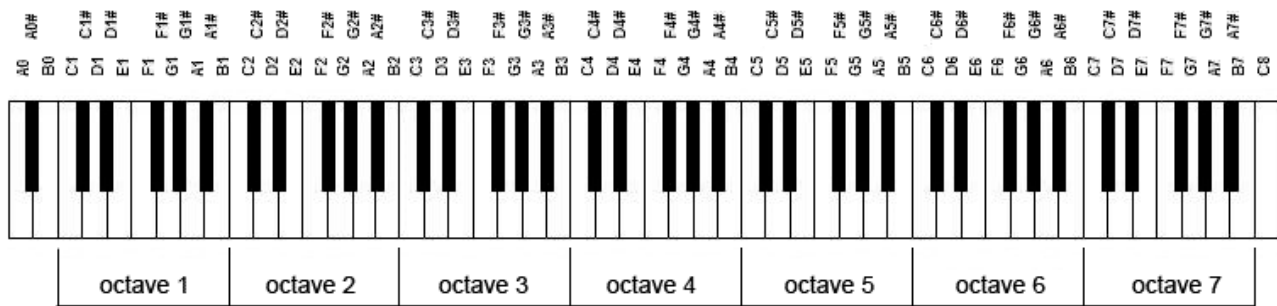


Figure 1 – Full Piano Keyboard

- **Pitch**

“Pitch, in music, position of a single sound in the complete range of sound. Sounds are higher or lower in pitch according to the frequency of vibration of the sound waves producing them.” - (Britannica, 2019)

- **Note**

“In music, a note is a symbol denoting a musical sound. Notes can represent the pitch and duration of a sound in musical notation. A note can also represent a pitch class. Notes are the building blocks of written music.” - (Nattiez, 1993)

- **Chord**

“Chord, in music, three or more single pitches heard simultaneously. Depending on the harmonic style, chords may be consonant, implying repose, or dissonant, implying subsequent resolution to and by another chord.” - (Britannica, 2013)

- **Octave**

“Octave, in music, an interval whose higher note has a sound-wave frequency of vibration twice that of its lower note.” - (Britannica, 2017)

- **Key**

“Key, in music, a system of functionally related chords deriving from the major and minor scales, with a central note, called the tonic (or keynote).” -

(Britannica, 2007)

- **Scale**

“Scale, in music, any graduated sequence of notes, tones, or intervals dividing what is called an octave. The interval relationships among pitches of a scale are its essential feature, and a particular pattern of intervals defines every scale.” -

(Britannica, 2017)

## 1.4 Research Questions and Objectives

The aim of this project is to successfully investigate the possibility of developing and fully implementing a Long Short-Term Memory based Recurrent Neural Network which is capable of composing music. By implementing the full machine learning pipeline, the true applicability of AI in the music industry can be put to the test.

The project itself meets the program's aims which I am studying by working with modern and current technologies which are applicable in this day and age in the job market. And the course itself has provided a new knowledge on the interdisciplinary field of Data Science by teaching me the relevant skills associated with this course and its modules which are essential for the completion of this project.

The following avenues of research have been identified:

- **Does Artificial Intelligence have a place in the domain of music?**
- **Is LSTM the appropriate approach to develop a network that is capable of generative models in music?**
- **Is the MIDI file format the most appropriate communications protocol for obtaining the tonal properties from an audio file?**
- **Are summative or formative based experiments the most appropriate for evaluating music?**

The following objectives have been identified:

- **To conduct a literature review on the applicability of AI in the world of music, an appropriate approach to development, and suitable methods to evaluate.** The literature review will consist of these subjects, with the intention of proving the validity of the thesis statement.
- **To design the architecture of the algorithm by identifying the high-level requirements necessary to achieve a successful result.** This step will consist of the functional requirements being identified, such as the basic features needed in the final product to determine it a success.
- **To create a dataset consisting of MIDI files and acquire the musical notation present in each individual track.** Note and chord objects will be identified using Music21 which is an open-source Python toolkit for computer-aided musicology. Thus, creating an output that the algorithm will be able to interpret.
- **To implement a robust algorithm based on the LSTM Recurrent Neural Network approach.** The algorithm itself must be capable of interpreting the musical notes present in the MIDI files (based on ABC notation), and train based on the sequences of notes identified. The final result will be an algorithm which is capable of producing a symphony of sorts on its own by generalizing from a certain amount of training performed on the train set of MIDI files.
- **To evaluate the integrity of the LSTM network and the final MIDI output by means of summative testing.** Evaluating your own work is crucial in identifying shortcomings upon reflection, and ways in which the project could be improved in future development.

## 1.5 Approach to Project Delivery

There are a vast number of frameworks and methodologies which may be acceptable for the developmental process of machine learning, but one that shines in particular amongst the rest is the agile approach, more specifically the Dynamic Systems Development Method (DSDM).

The reason that this methodology is superior to many others is due to the high level of flexibility that it offers, enabling developers to iteratively shift between the various developmental phases. Which is crucial in allowing failure to occur as fast as possible so that it can be remedied, and lessons learnt from the failed outcomes.

Instead of a methodology such as the waterfall approach which could result in a terrible outcome in a project such as this which requires repetitive phases of development, troubleshooting, and testing. The waterfall approach offers a linear path from the point of inception till final product launch, which ultimately means that a team of developers must continually progress forward and cannot at all repeat steps in the process such as testing, and then continuing developing.

The machine learning pipeline is an intricate process which is prone to debugging and constant troubleshooting to find the route of specific problems that will occur, and the organic flow that DSDM promotes is perfectly suited to this scenario, where going back over previously done work is encouraged, to ultimately ensure that the best product is achieved absolutely bug free and ready for deployment.

The nature of the methodology allows for this as things are done step by step in an iterative motion thus allowing for the most important thing to always be implemented next in a high-level specification prioritisation. Building incrementally from firm foundations is the core of this approach.

## **2 Literature Review**

### **2.1 Approach to Literature Searching**

The purpose of the literature review is to gather information based on the existing research on the topic in order to gain an understanding. This is achieved by conducting said investigation into several avenues consisting of research reports, articles, published papers and journals. Through thorough analysis, it is made clear how others in the field have attempted to tackle similar problems as the one faced in this scenario, and with this knowledge it is possible to contribute new information to the field by avoiding the potential errors that others may have made, or simply improve upon them and/or learn from them to obtain a greater understanding.

With all of those factors taken into consideration, an optimal path forward is created where important and relevant information is at the forefront of this section of the report, reinforcing the thesis itself and paving the way to a successful implementation which is backed by the facts established in this section.

### **2.2 Identifying the Problem**

Problem characterization is tackled by obtaining a sufficient understanding in the relevant subjects pertaining to the topic. This begins with initial research onto the broader subject matter itself, which is artificial intelligence in the domain of music. In order to comprehend its relevance and the history of the progress that has been made in the field this preliminary form of research on the broader subject matter is essential in creating an understanding of the evolution that has taken place over many years of progression in the field.

It is only once this understanding has been developed that more concentrated and refined research can be conducted on the specifics pertaining to the details of the research which will ultimately reinforce the thesis. From the insight gained through this research, a concrete foundation is put in place that allows the goals of the project to be determined in a decisive manner. The aforementioned goals are further reinforced by the detailed

and technical aspect of the literature review which speaks of the sophisticated technologies used in the implementation and why they are an appropriate choice.

## **2.3 Artificial Intelligence in the domain of Music**

From 1957 until the early 70's is where the history of AI and music began. There was absolutely no pre-existing data on the subject matter, and as a result the primary focus of research was conducted on algorithmic composition, which is the technique of using an algorithm to create music. In layman's terms, basically refers to "the process of using some formal process to make music with minimal human intervention" - (Alpern, 1995)

The reason that focus was shifted toward machine learning techniques for music composition is because of its nature to produce something which isn't necessarily predictable, thus adding an element of creativity and uniqueness that can result in a melodically complex sound being generated by an algorithm. And the perfect facilitator for this was a long short-term memory based neural network which is capable of processing entire sequences of data instead of single data points.

It wasn't until the 80's that genuine breakthroughs were made on the subject matter, because this is when more focus was put toward understanding the sequences which define music and algorithms capable of building upon pre-existing music. It was at this time that the methodology of generative modeling was created and adapted to this specific use case. A generative model includes the distribution of the data itself, and determines how likely a given example is, this is particularly useful on the subject of music composition achieved by AI because it can assign a probability to a sequence, therefore predicting what should come next, and music is completely comprised of sequences.

It was in 1980 when David Cope from the University of California developed the software EMI, which is Experiments in Musical Intelligence. The system he implemented was based on generative models that were used to analyze existing music and from that basis, create new musical pieces.

Currently research on the subject matter is being conducted by many of the world's leaders in technology such as Google, IBM, and Sony. Into various avenues such as the cognitive science involved with the way the brain perceives music, intelligent sound analysis, and music composition with little to no human intervention.

The research and implementation conducted in this project builds upon existing work and adds a new dimension to the field because currently the only way a regular person can access software capable of music composition is by purchasing highly sophisticated and polished software that gives absolutely no insights to the inner workings of the software itself. It is simply a GUI interface with a few options to generate music based on specific genres. But what is being developed in this project is the open-source full pipeline that can give a potential user the insights and understandings of the algorithm itself, with the ability to input a unique dataset and create their own sound.



## 2.4 ANN's and their Perspective Architectures

“Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.” - (IBM, 2020)

### 2.4.1 Neural Networks

There are several unique layers which encompass a neural network, defining what it is and the way in which it functions. A collection of connected nodes called artificial neurons is what neural networks are based on. These nodes consist of several layers being an input layer, several hidden layers, and an output layer. There are connections which are formed between each node, and each node has its own weight which is the parameter within a neural network that transforms input data within the network's hidden layers.

Regarding the use cases of neural networks, the most ideal scenario is on the subject of classification issues. NN's possess the capability to adapt according to the various types of data which are inputted, thus making them adaptive. They also retain an incredible level of flexibility concerning its applicability to real-world scenarios, particularly when it comes to modelling complex solutions to do with prediction, image recognition, or natural language processing.

However, there are various different types of NN's, and each has specific features and capabilities which define them from the rest. The two comprehensive categories which encompass the large majority of all NN's are Recurrent NN's (RNNs) and Feed-Forward NN's (FNNs). “These models are called feedforward because information flows through the function being evaluated from  $x$ , through the intermediate computations used to define  $f$ , and finally to the output  $y$ .” - (Gupta, 2017)

There are no feedback connections in which outputs of the model are fed back into itself and that is what truly defines the FNN from a RNN. This is also the very factor which makes a FNN inappropriate in the use case surrounding music generation, because no past state of the neural network can affect the future states of the weights within the network. This lack of memory creates an inability to handle any data type which primarily consists of sequences, the reason being that sequences such as music rely on a recollection of past notes which are essential in order for generalisation to occur when an attempt is being made at generating new notes that are both coherent and consist of a firm musical structure.

### **2.4.2 Recurrent NN's**

With FNN's being ruled out in terms of aptness for this particular scenario, the next broad category to consider regarding this use case are RNN's. The hidden layers within a Recurrent Neural Network have intermittent and recurring connections which exist between previous and current states within a neural network, and what this ultimately results in is information that acts as memory because it is stored in the form of activations.

The applicability of this neural network has relevancy on the subject of memory related scenarios such as music composition due to its capability of modelling short term dependencies. But the key issue with this approach is the fact that this type of neural network only stores information regarding the previous state, that is the full extent of its memory due to the nature of the activation layers. When analysing music there is always a beginning, middle and end, and key features from any stage of this process often hold relevance in an alternate stage, for example, a potential introduction to a song might contain a melody which repeats throughout the bridge of the song that will lead into its chorus.

Considering this, only storing information from the previous state removes a core aspect of music which is creativity, and turns it into a sequence of pure structure. All of the previously learned information needs to be considered before moving onto the next increment within the training process, and this rules out RNN's as an optimal means of music generation.

### 2.4.3 LSTM NN's

Long Short-Term Memory Neural Networks, otherwise referred to as LSTM NN's, solve the issue which was discussed in relation to RNN's because a unique cell exists within the LSTM network, and this is the memory cell.

“These memory cells consist of three gates: input, output and forget. Input gates control how much data is inputted into memory, output gate controls the data passed to next layer and the forget gate controls the loss or tearing in the stored memory” - (Tch, 2017)

The regulation of incoming and outgoing data is handled by the values stored within the gate, and the function of this is to protect the memory cell value. When data is no longer needed and can be disposed of to clear up more memory for the next part of a sequence, the forget gate can be used. The values stored within the memory cells of a LSTM NN decay exponentially, and this is entirely due to the nature of the forget gate.

The action of previous data being forgotten when it is no longer needed is extremely crucial in the process of a LSTM NN due to the network having a finite memory capacity, and the most recent information being input to the network is typically more valuable than the older information, but newer information being more relevant does not render the older information irrelevant, there is a perfect balance which must be maintained in the sense of establishing a decay rate, so that older important information is retained without creating biases within the network due to too much old information being utilised that isn't particularly useful.

The network itself establishes importance of information and is able to read, write and delete said information when deemed necessary. This feature aids in the recognition of note sequencing and the processes involved with melody structure by utilising the full capabilities of the Long-Short Term Memory Cell.

## 2.5 Methods for Evaluation on Generative Models in Music

The desire to truly understand what defines creativity has led to the creation of computationally creative systems which exist amongst a substantial diversity of different use cases. Spanning from image and speech generation, all the way to music generation. This constant drive that exists is continuously pushing and redefining the bounds of what can be achieved through machine learning methodologies, and with these continuous advancements comes new challenges in the sense of procedures for evaluation.

“We review the evaluation of generative music systems and discuss the inherent challenges of their evaluation. Although subjective evaluation should always be the ultimate choice for the evaluation of creative results, researchers unfamiliar with rigorous subjective experiment design and without the necessary resources for the execution of a large-scale experiment face challenges in terms of reliability, validity, and replicability of the results” - (Yang, 2018)

On the subject of image generation the procedure used for assessment utilizes a pattern recognition model which in turn evaluates the generated sample. This methodology is known as the Inception Score.

“The Inception Score, or IS for short, is an objective metric for evaluating the quality of generated images, specifically synthetic images output by generative adversarial network models.” - (Brownlee, 2019)

When using the Inception Score a general assumption is made, and it is that a human-like classification ability is present in both humans themselves, and well-trained image classifiers. And although the findings from this method of evaluation seem assuring, there is a lack of scientific evidence which points to the direct correlation between the capability of human judgement, and that of the evaluation methodology. Therefore, leaving an open question in the midst of a solution.

### 2.5.1 Approach

Particular importance was drawn toward a methodology surrounding the means of evaluation for image generation in order to highlight the key differences between that of image generation and music generation, and how these differences result in greater challenges being faced in the latter.

Even greater challenges are faced on the subject of music generation because of the various dynamics enveloping music. Beginning with the connection between emotion and music which is an age old question, followed by the specifically structured yet sequential nature it possesses, making for a uniquely difficult experience when navigating through various means of evaluation. It is for these reasons that the automation of computational models have not been successful in meaningful evaluation.

The generative approach to developing music suffers due to the inability of proper evaluation methodologies. And as a result, this method for developing music with the assistance of machine learning is not sophisticated enough to compete with the most current solutions. There are two primary challenges which researchers are faced with when trying to assess models which were developed with the generative approach. Thus, being summative and formative assessment methodologies.

The former being a method in which a subjective approach is taken in the sense of evaluation through listening-based experimentations, as a result of this form of assessment many biases can be formed which lead to inaccurate results. Formative, being the latter of the two challenges is an objective based approach, which can often lack relevance.

## 2.5.2 Summative

On the subject of summative/subjective methodologies for evaluation, there is an approach which slightly modifies the Turing Test by adapting it toward music generation.

“The Turing Test is a method of inquiry in artificial intelligence (AI) for determining whether or not a computer is capable of thinking like a human being. The test is named after Alan Turing, the founder of the Turing Test and an English computer scientist, cryptanalyst, mathematician and theoretical biologist.” - (St. George, 2019)

The way in which this method of evaluation is adapted is by asking the test subjects to distinguish which pieces of music they believe were composed by a human as opposed to an algorithm or computer. Although this method has seen success, one of its shortcomings is that even if a subject is able to distinguish between which piece was developed by a human or computer, this does not attest to how aesthetically pleasing the sound was, once again posing a question amidst a solution. More questions in relation to this evaluation method exist, such as how musically trained were the subjects, if at all, or the quality of the listening environment.

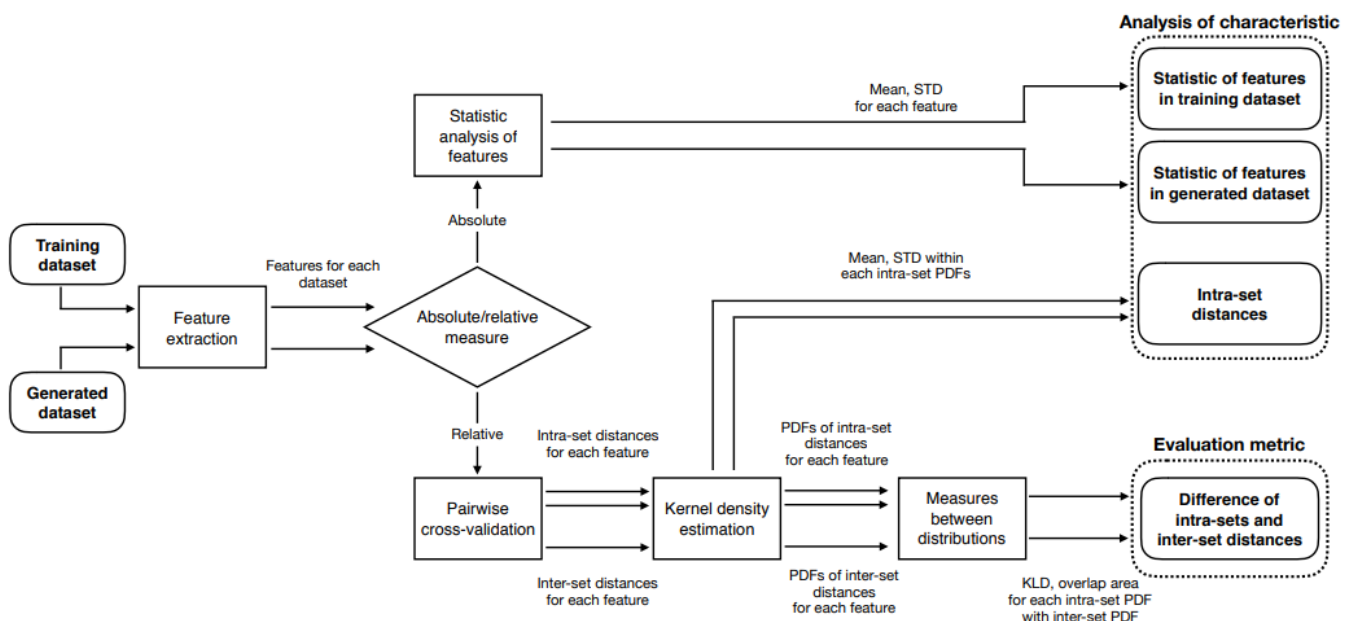


Figure 2 - Proposed workflow of the formative approach to evaluation - (Yang, 2018)

### 2.5.3 Formative

On the subject of formative/objective methodologies for evaluation, these are considered to be more acceptable in relation to assessing such experiments due to the reproducibility, reliability and validity of said experiments.

When addressing the various evaluation methods for a model designed for generative music creation, it is important to consider the multi-criteria nature that it possesses in order to design the optimal testbed. With musical domain knowledge it is possible to establish a series of tests which are understandable to both humans and computers, which comprise of performance metrics that are both indicative of essential criteria relating to the structural integrity of the music and its processes, as well as how aesthetically pleasing it may sound which perfectly coincides with the previously established metrics and how they relate to the actual sound of music in a real-world scenario.

These criteria fall into various music related categories such as rhythmic patterns, tonal distance, pitch classes, interval frequencies, etc. When all of these attributes are analysed as one, it is possible to determine how they contribute to an aesthetically pleasing sound, drawing one step closer to understanding how creativity, music, and emotion coincide.

The designs of both a listening based experiment and metrics based on domain knowledge are complex, therefore leading to both forms of evaluation having a list of their pros and cons, which ultimately may result in failure. Either due to there being an insufficient amount of resources for a subjective based approach, or because the objective based approach is too heavily concentrated on structures which deviate from musical rules.

## 2.6 Review of Related Work

In recent years a number of deep neural networks have been implemented with either the hopes of introducing a new approach to tackle the task, which is machine learning based music generation, or to add new findings to pre-existing work that either builds upon it or discredits it.

This section of the literature review aims to identify pre-existing work on the subject matter whilst simultaneously comparing and contrasting it against what has been achieved in the implementation that supports this thesis.

An approach that similarly made use of the LSTM methodology to music generation was implemented by Christian J. Walder and Dongwoo Kim, in which they recognized that motifs are typically repeated and transformed throughout music and capitalized upon this factor by designing a neural network that could successfully identify motifs throughout a piece of music and their transformations that were carried out within the sequencing of notes.

This was achieved via the concept of motif transposition and the model was based on a learned edit distance mechanism which generalizes a classic recursion from computer science, leading to a neural dynamic program.

“A motif is a small but recognizable musical unit. The motif might consist merely of a series of pitches or a distinctive rhythm, or it might be harmonically conceived; quite often, pitch and rhythm are combined in a motif to create a discrete melodic fragment.” - (McLamore, 2021)

What this approach led to is a model that was highly sophisticated in recognizing the regularities in a sequence of music, with a significantly higher success rate than typical LSTM based approaches. The capability to produce more structured melodies makes this approach desirable due to its unique ability to identify the motifs in a piece and generalize based on them, but the shortcomings of this approach is that it strictly works on symbolic music, similarly to what has been implemented in this thesis, meaning that



the algorithm can only function with an input stored in a notation-based format such as MIDI files, which contain explicit information about note/chord onsets and pitch.

A more sophisticated approach to solving this issue that could grant the network more dexterity would be to implement custom encoding as was achieved by Nabil Hewahi et al. (2018), in which “the complexity of the model was increased by using a sliding window to capture given notes within a time interval without making a distinction between single notes or chords.”

The key benefits of this approach to handling MIDI files that was implemented by Nabil Hewahi et al. (2018) is that by creating their own form of custom encoding to handle the MIDI files, they were able to introduce varying note and chord durations, which adds a significant amount of depth and cadence to a music piece, therefore solving the issue of fixed time intervals and offsets that are seen in many other implementations, which can potentially be viewed as inhibiting a piece from sounding as human-like as possible.

## 2.7 Conclusions

It has been clearly established that artificial intelligence has a valid place within the domain of music, and the research and studies conducted into the subject matter have aided in furthering developments which have contributed toward a greater understanding of the processes and structures which define music, and unique ways of interpreting them.

These advancements aid not only musicians, but also the expansion and progression of machine learning, by conducting sufficient tests which determine the genuine dexterity of these algorithms and how they can be applied to real-world scenarios in which people and AI co-exist in a manner which is beneficial to both parties.

The most appropriate way to continue these advancements has also been established, by careful analysis of the various technologies available for the implementation, with the LSTM NN based algorithm shining above the rest due to its unique ability to store previously learned information in a cohesive manner which is accessible to itself, in order to perfect the training process and output a sound which is the result of training coherently.

In terms of identifying an appropriate method for evaluating such a neural network, the two major avenues that were considered and analyzed in depth are summative/subjective, and formative/objective.

When designing an experiment that takes into consideration the various critical factors relating to the way a musical piece sounds such as rhythmic patterns, tonal distance, pitch classes, interval frequencies, etc, it is deemed possible to determine how they contribute to a musically accurate sound in terms of its structure, therefore this method of evaluation with a formative basis potentially lacks relevance when evaluating a musical piece.

Laslty, when reviewing related work it has been determined that there are several pre-existing implementations on the subject matter, and the most sophisticated of them all

make usage of the LSTM neural network as its basis, with key distinctions being in the way MIDI files and the data pertaining to them is interpreted and then utilised in the network itself. Contributing largely to how complex the sound being output is, such as an unusual degree of musical dissonance on one end of the spectrum, or to how tonally complex the harmonies are on the other.

### **3 Legal, Social and Ethical Issues**

In terms of legal issues associated with this project, the key factor that needs to be taken into consideration is Intellectual Property Rights (IPR). Stealing another individual's idea which has already been created can breach copyright law as it is theft of intellectual property. In this particular scenario the concept being implemented is not any person's intellectual property as it is merely a concept. Although some individual person's and/or organisations have done similar implementations which they launched as their own unique software, the software itself is protected as intellectual property, but the idea is not.

Another legal issue to take into consideration in this scenario is Copyright Law because the project handles the subject of music. The music itself which will be fed into the algorithm for training purposes is publicly available and can be used at one's discretion. Copyright Law specifies that as long as there is no distribution involved, then an individual may use it how they please, and for the purpose of this project, as stated previously, publicly accessible music will be used (most likely from a game soundtrack). And the algorithm will interpret the musical notes present in the soundtrack and train according to this. Ultimately, what the algorithm outputs is a unique sound and so the project does not breach any laws as nothing that is protected by Copyright Law will be distributed.

As briefly expressed previously, the topic of AI in the music industry is controversial in the eyes of some members of the industry. But it is something that needs to be discussed as technology continues to advance at a rapid pace.

Change due to automation is an occurring phenomenon which has been seen in many industries, and now it has entered the world of music.

The reality of the situation is that musicians need to adapt to technological advancements, because the goal is not to replace musicians with AI, it is to integrate and achieve something never accomplished before. For the foreseeable future, AI will always be dependent on humans for creative input which is the foundation of music.

A potential ethical issue which may arise if the project is ever launched and made publicly available is a matter which borders the legality of intellectual property. Prospective users who may use the algorithm to create their own music might intend to claim it as their own and even move on to sell it, meanwhile it was my intellectual property which was used to output the symphony which they created.

This is something that would need to be specified in an End-User License Agreement, which is a legally binding contract entered into between a software developer and the user of the software. This documentation would specify if a prospective user were free to develop their own symphonies using the algorithm, or if they may potentially need to pay royalty fees on any money they may earn from selling symphonies generated using the algorithm, for example.

## **4 Design**

### **4.1 Introduction**

This section of the report is essential in outlining the way in which every aspect of the implementation will be achieved. By identifying the key concepts that must be accomplished, a concise image of the final outcome is produced, and this can only be done by bringing said concepts together.

The planning which had been done prior to the commencing of product development will be showcased in order to demonstrate a high-level understanding of the milestones which have been accomplished and when. Followed by an identification of the system requirements, and what needed to be done to achieve a minimum viable product that demonstrates a proof on concept.

Finally, ending with an overview of the system architecture, explaining the hardware and software packages that were necessary to succeed during the implementation phase, and why they were crucial to the success.

### **4.2 Initial Planning**

Initial planning began by identifying the various deliverables that exist within the project and what sub-categories they fall within, once the deliverables had been identified time constraints could be estimated in order to establish a realistic idea of how long each requirement would take to be fulfilled.

| <b>Research:</b>                            |   |   |                               |
|---|---|---|-------------------------------|
| Initial Proposal<br>[Time = 7 days]         | Initial Report<br>[Time = 7 days]               | Interim Report<br>[Time = 14 days]                    | <b>Total Time:</b><br>28 days |
| <b>Analyse:</b>                             |   |   |                               |
| Analysis of Research<br>[Time = 3 days]     | Analysis of Technologies<br>[Time = 5 days]     | Analysis of Project Deliverables<br>[Time = 1 days]   | <b>Total Time:</b><br>9 days  |
| <b>Design:</b>                              |   |   |                               |
| Problem Characterization<br>[Time = 3 days] | Requirement Analysis<br>[Time = 5 days]         | Statement of Requirements<br>[Time = 7 days]          | <b>Total Time:</b><br>15 days |
| <b>Build:</b>                               |   |   |                               |
| Algorithm Prototyped<br>[Time = 14 days]    | Algorithm fully implemented<br>[Time = 14 days] | White box Testing<br>[Time = 7 days]                  | <b>Total Time:</b><br>35 days |
| <b>Evaluate:</b>                            |   |   |                               |
| Evaluation of Report<br>[Time = 1 days]     | Evaluation of Product<br>[Time = 1 days]        | Evaluation of Personal Performance<br>[Time = 1 days] | <b>Total Time:</b><br>3 days  |
| <b>TOTAL TIME TAKEN FOR ENTIRE PROJECT:</b> |   |   |                               |
| 90 days<br>Or<br>2.9 months                 |   |   |                               |

**Table 1 - Time Constraints**

Once this basic understanding of the potential time constraints had been created, it was then made possible to create an actual project timeline with ideal dates in place that outlined by when each deliverable should be achieved.

| Activity | Start      | End        | June   |        |        |        | July   |        |        |        | August |        |        |        | September |        |  |  |
|----------|------------|------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----------|--------|--|--|
|          |            |            | 09-Jun | 17-Jun | 25-Jun | 30-Jun | 09-Jul | 17-Jul | 25-Jul | 31-Jul | 09-Aug | 17-Aug | 25-Aug | 31-Aug | 09-Sep    | 16-Sep |  |  |
| Research | 01/06/2021 | 28/06/2021 |        |        |        |        |        |        |        |        |        |        |        |        |           |        |  |  |
| Analyse  | 29/06/2021 | 08/07/2021 |        |        |        |        |        |        |        |        |        |        |        |        |           |        |  |  |
| Design   | 09/07/2021 | 24/07/2021 |        |        |        |        |        |        |        |        |        |        |        |        |           |        |  |  |
| Build    | 25/07/2021 | 29/08/2021 |        |        |        |        |        |        |        |        |        |        |        |        |           |        |  |  |
| Evaluate | 30/08/2021 | 03/09/2021 |        |        |        |        |        |        |        |        |        |        |        |        |           |        |  |  |

**Table 2 - Project Timeline**

Slight deviations from the original planning model have occurred due to various setbacks, but for the most part work has coincided relatively well with the timeline, thus enabling optimal workflow which ultimately leads to the greatest results.

Planning was based on various factors such as real-life commitments, pre-existing knowledge on the problem itself at hand, and new information gathered that is partly made up of the research which has been conducted in form of the literature review, as well as research into the various methods which can be used to achieve the implementation itself.

### **4.3 Analysis of Requirements**

As previously described, problem characterization is tackled by obtaining a sufficient understanding in the relevant subjects pertaining to the topic. This statement sits at the crux of this section of the report because in characterizing the problem at hand, a greater understanding of the relevant disciplines is made clear, therefore procuring the obvious path forward which encapsulates the entire problem scenario and the key functionalities that should exist within the system to achieve a viable product.

The aforementioned key defining functionalities are further broken down into MoSCoW prioritization so as to attain a more finite understanding of the appropriate task ordering that should occur.

### 4.3.1 Functional Requirements

| Entity             | Requirement   | MoSCoW Prioritization |
|--------------------|---|-----------------------|
| System/Environment | Create a network that can successfully predict on the training set. | M                     |
| System/Environment | Generate notes from the trained network.                            | M                     |
| System/Environment | Input a dataset consisting of MIDI files.                           | M                     |
| System/Environment | Convert notes/chords into binary format.                            | S                     |
| System/Environment | Make use of a LSTM neural network.                                  | S                     |
| System/Environment | Evaluate the final output.  | S                     |

Table 3 - Functional Requirements

### 4.3.2 Non-functional Requirements

| Entity             | Requirement  | MoSCoW Prioritization |
|--------------------|--|-----------------------|
| Usability          | The algorithm must meet certain criteria that determine the project a success.     | M                     |
| Maintainability    | The code written for the algorithm must be maintainable, updatable, and efficient. | M                     |
| Required Resources | The algorithm itself must be trainable and functional on the system.               | M                     |
| Platform           | The algorithm must be compatible with different software versions.                 | S                     |
| Readability        | The code should be readable and understandable.                                    | S                     |
| System/Environment | The trained model weights should be savable.                                       | C                     |
| System/Environment | The trained model loss value should be calculated.                                 | C                     |
| Response Time      | The software should have fast response times whilst in use.                        | C                     |

Table 4 - Non-functional Requirements



## 4.4 Overview of System Architecture

An overview of the system architecture comprises of two critical factors, the software/packages and the hardware required to transform an idea into a prototype. As well as the conceptual model itself that encapsulates and defines the structure and behavior of the system, reinforced by appropriate descriptions and an approach to organization that supports reasoning about the aforementioned structures and behaviors of the system.

### 4.4.1 Software

All of the software used during this project is open source and readily available for anyone who desires to access and make use of it. This ranges from the operating system of the local machine in use, all the way to the platform which will be used for the final implementation. Anaconda is a distribution of the Python programming language which will simplify package management, ultimately enabling the development of this project.

On the local desktop in use for development, the operating system in use is Ubuntu 20.4 which works natively with Python, the programming language used to implement this project. CUDA version 10.1 and TensorFlow version 2.2.1 had already been setup prior to the commencing of the implementation and were fully functional and prepared for development to begin.

| Python Packages used for Development via Jupyter Notebook |  |  |
|---|--|--|
| Package:  | Description:   | Use:   |
| Glob  | “The glob module is used to retrieve files/pathnames matching a specified pattern. The pattern rules of glob follow standard Unix path expansion rules.” | Glob is used to retrieve the MIDI files which make up the dataset by importing them before they can be parsed.   |
| Music21   | “Music21 is a Python-based toolkit for computer-aided musicology.”   | MIDI files are parsed into Music21 stream objects, and in this format the software is able to obtain all of the note/chord objects present in each file. |

|            |  |   |
|------------|--|---|
| Pickle     | “Pickle is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database.” | Once each individual MIDI file is separated into note/chord objects, the data is dumped into a pickle file and is then ready for interpretation by the neural network.  |
| Numpy      | “NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices.”   | Numpy is used to reshape the input (notes obtained from MIDI file) into a format compatible with LSTM network layers.   |
| TensorFlow | “TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.”       | Both TensorFlow and Keras integrate together. This symbiotic relationship operates by allowing Keras to run on top of TensorFlow, providing a high-level interface for communication. The neural network library that is responsible for providing the LSTM layers and training the model itself are a result of the aforementioned high-level API provided by Keras. |
| Keras      | “Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.”                                       |   |

**Table 5 - Packages Required**

#### 4.4.2 Hardware

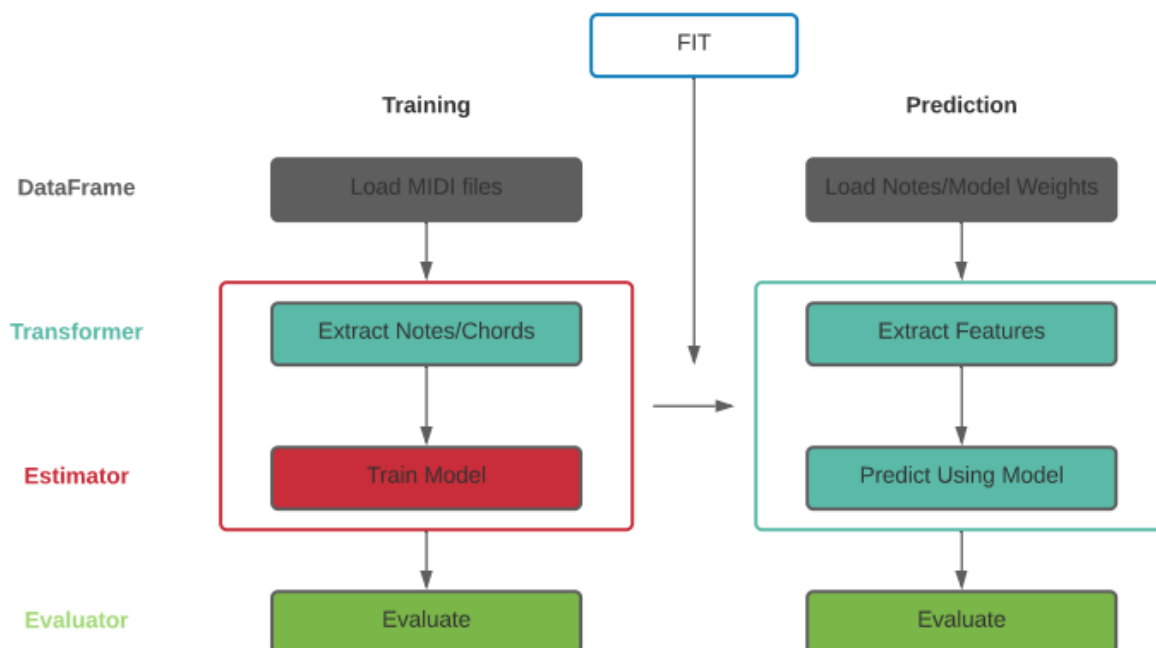
Specific hardware with a certain amount of processing and graphical power is essential in the efficient training of Machine Learning models. The local machine used during development for this project consists of specifications which have been listed below, which make it more than capable of training models in an efficient manner.

It is preferable to make use of a GPU (Graphics Processing Unit) whilst training models because of the high bandwidth offered, which masks the latency under thread parallelism, thus rendering it a much more efficient method for accelerating the process of training.

| Local Desktop Specification: |                       |                            |
|------------------------------|-----------------------|----------------------------|
| Category:                    | Component:            | Details:                   |
| Operating System             | Ubuntu                | Version 20.04              |
| Central Processing Unit      | AMD Ryzen 7 5800x     | 3.8 GHz 8 cores 16 threads |
| Graphics Processing Unit     | Nvidia RTX 2080       | 8GB GDDR6 Memory           |
| Random Access Memory         | 16GB DDR4             | 3866 MHz @ CL15-15-15-30   |
| Primary Storage              | Sabrent 1 TB NVME SSD | PCIe-3.0                   |

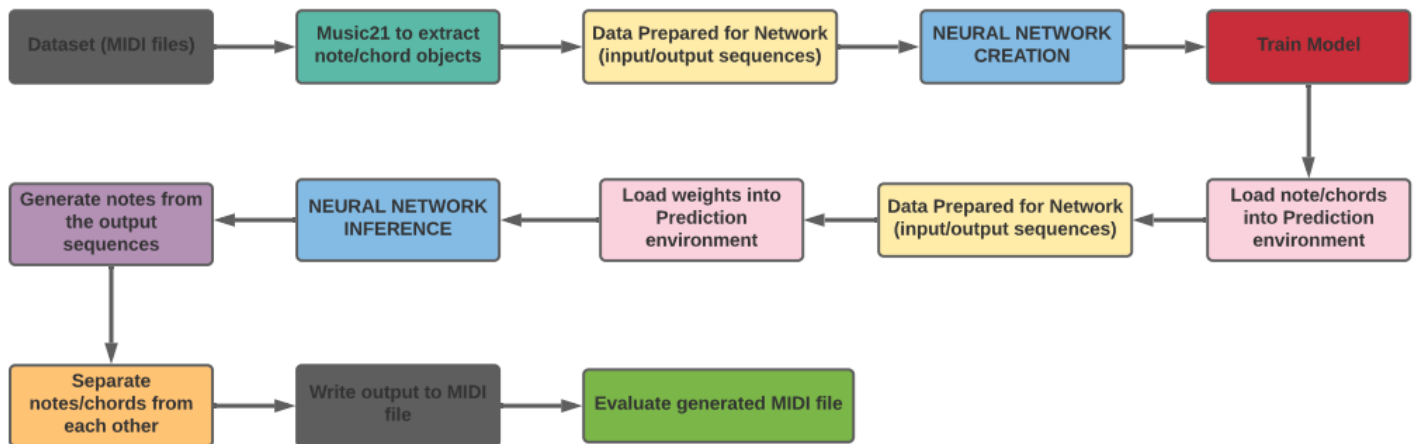
**Table 6 - Hardware Specifications**

### 4.4.3 Proposed Model Architecture



**Figure 3 - Pipeline Workflow**

#### 4.4.4 Proposed Model Workflow



**Figure 4 - Pipeline Architecture/Process**

The above diagrams have been provided above to demonstrate the proposed model workflow and architecture that will exist within the implementation. The purpose of these diagrams is to create a basic understanding of the precise workflow process behind the algorithm itself with the intention of highlighting the cyclic nature of the pipeline that isn't necessarily a one-way flow system. This previously mentioned cycle nature enables an iterative approach to the machine learning pipeline itself, thus resulting in outcomes which may be improved and a more scalable model.

The diagram itself encapsulates the entire proposed system by showing the two separate environments that will house the implementation. Thus being, a training environment and a prediction environment. The core functionality behind each separate environment is represented in the square boxes. The features extracted from the training environment will be fit to the prediction environment, where a final outcome will be generated and then assessed.

To describe the steps of the proposed architecture in layman's terms, the workflow process begins with MIDI files being loaded in via the glob package, these MIDI files are then parsed into Music21 which is the software package responsible for extracting the note/chord objects from the files, the now extracted note/chords are placed inside a

file and converted into binary format for the purpose of easy interpretation on behalf of the neural network, the files are reshaped and normalized and finally input to the network. The LSTM network trains based on the contents of the dataset and the weights are saved. The trained model will then be evaluated for its loss against the test set.

The second part of the system is prediction. The previously created files containing the notes/chords and the model weights are loaded in and their features are extracted to a newly created LSTM network for inference, a function then defines the sequences which the network must generalize on, and the network predicts an output, the final output is then converted to a MIDI file which can be evaluated.

## 5 Implementation

### 5.1 Introduction

This section of the report is critical in explaining the various steps that were taken and decisions that were made to achieve the final product that is a result of a successful implementation which demonstrates the capabilities of a LSTM neural network, and its validity in generative based approaches when creating music, which capitalize on sequence-based data.

The following section will contain a succinct and explicit explanation of the entire process which led to a successful implementation, ranging from why this specific dataset was chosen, through to training a model, then fitting the extracted features to a network that predicts the next sequence of notes to come.

### 5.2 Training Data

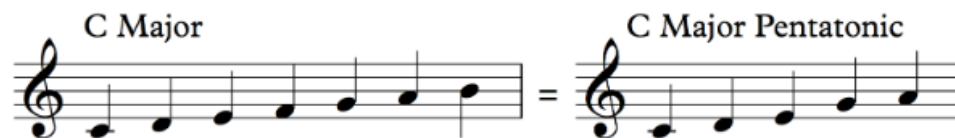
For the experimental phase of this project the dataset was created from the soundtrack of various Pokémon games, which contain beautiful music and piano melodies consisting of elements of jazz, rock, and jazz fusion, to create an utterly unique sound which has captured the hearts of the fanbase.

The file format of the dataset is MIDI, and the justification for creating a dataset consisting of MIDI files over any other format is due to the nature of the MIDI file and how instructions are stored within it on how to create the sound, thus allowing access to the tonal properties of a file, and the way in which they are accessed in this project is with the use of Music21 which is a Python toolkit consisting of package imports that serve as a connection to the musicology software which is Music21, that possesses the ability to obtain the aforementioned tonal properties from each individual MIDI file.



Figure 5 - Excerpt from a MIDI file within the dataset

Once the data was parsed through Music21 it was possible to determine a few key features about the tonal properties of the dataset. The first of which being that the MIDI files consist of a 4-bar structure within a 4/4 time signature, meaning that there are 4 quarter beats in each measure, with a typical stepsize of 8 notes per bar, and a note offset of 0.5, meaning that the time taken from the start to end of an individual note is typically 0.5 seconds.



**Figure 6 – Heptatonic versus Pentatonic**

Most of the music within the MIDI files is written in C Major which is a commonly used scale within this very specific genre of music found in the Pokémon games and their soundtracks. As a result, the C Major heptatonic scale is what's used for improvisation by the neural network during the prediction phase due to the basis of the training data being structured around this musical scale.

The typical heptatonic scale within music consists of 7 notes per octave, which was described in Music Background Information, but the pentatonic scale of any given note consists of only 5 notes per octave, therefore the standard C Major heptatonic scale is used instead of its pentatonic variant due to the fact there are more notes for the network to interpret and generalize based on.

## 5.3 Dataset Preparation

The first step in dataset preparation was deciding on an appropriate data format, that could synergize optimally with the approach being taken in the sense of needing access to certain attributes from each file. The obvious decision was to utilize the Musical Instrument Digital Interface format, otherwise known as MIDI.

Within MIDI files message instructions are stored that contain explicit information about note/chord onsets and pitch, duration, volume, etc. What makes this so different to other formats such as MP3 is that audio isn't stored within the file, instead it is a set of instructions that explicitly define how to produce sound based on the various attributes stored within the file itself.

The nature of MIDI files is critical to the integrity of this project because the attributes pertaining to each individual MIDI file are accessed by both Music21 and MuseScore, which are musicology-based platforms for the interpretation of MIDI files. Music21 is responsible for the extraction of note/chord objects from the file itself, and MuseScore is responsible for translating MIDI files into editable sheet music which allows the musical structure to be properly analyzed.

With the use of the Music21 package within the environment, the note and chord objects are obtained and within them, specific information about the pitch, octave and note offset, as was mentioned previously and these concepts were described.

```
try: # file has instrument parts
    # Separating the different sounds present in each midi file
    s2 = instrument.partitionByInstrument(midi)
    notes_to_parse = s2.parts[0].recurse()
except: # file has notes in a flat structure
    notes_to_parse = midi.flat.notes

# isinstance returns true if criteria is met, otherwise returns false
for element in notes_to_parse:
    if isinstance(element, note.Note):
        notes.append(str(element.pitch))
    elif isinstance(element, chord.Chord):
        notes.append('.'.join(str(n) for n in element.normalOrder))
```

**Figure 7 - Separating the note/chord objects in each MIDI file**



In preparation for the neural network, the entire dataset is placed inside an array and then parsed into Music21 where it is converted to a stream object. Music21 handles MIDI data like this because in that format it is able to obtain the aforementioned key attributes pertaining to each individual MIDI file.

The string notation of each note has its pitch appended to it, and every chord is appended by encoding the identification number of each present note that exists within the chord into an individual and separate string. The notes within each chord are then separated by a punctuation mark which is otherwise referred to as a full stop. The purpose of this is to simplify the decoding process with the output generated by the network.

The notes and chords obtained through this process are placed into a sequential list and are finally dumped using pickle into binary format at the end of the training process, in preparation for interpretation by the prediction-based neural network.

## **5.4 Pipeline Development & Feature Extraction**

### **5.4.1 Mapping Function**

The next phase which occurs in preparation for the neural network is the preparation of sequences that will serve as the network input.

One of the most critical aspects of this project that need to be taken into consideration for the implementation is the way in which the neural network will be able to interpret the notes/chords from the dataset. As LSTM specializes in handling sequences of data where patterns occur that can be generalized on, integer-based numerical data is essential for its input. And as a result of having to deal with string-based note/chords, a mapping function is created in which the aforementioned string-based categorical data is converted to integer-based numerical data.

```

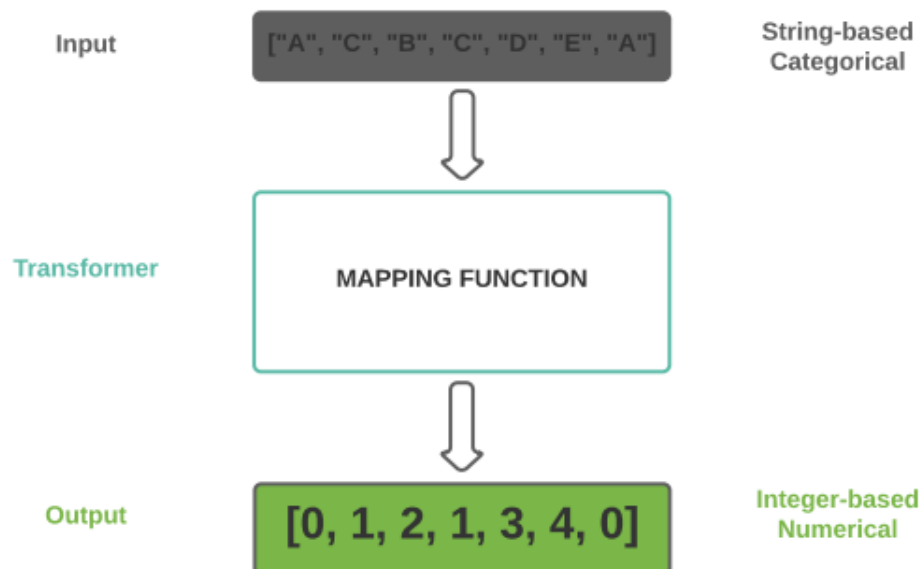
### MAP FUNCTION ###
# create a dictionary to map pitches to integers
note_to_int = dict((note, number) for number, note in enumerate(pitchnames))

```

**Figure 8 - Mapping String-based Categorical to Integer-based Numerical**

The way in which this mapping function works is by converting the note/chords into integer indexes. And this is achieved by identifying distinct values and then representing them with an integer-based counterpart which is a representation of where the unique distinct value is positioned.

In layman's terms, if the note A is the first to appear within the dataset, the mapping function will map this distinct value to 0. Any other occurrence of the note A which may be present elsewhere in the dataset will always be recognized as 0. Following on this basis, if the second note to occur within the dataset is B, as it is a distinct value occurring for the first time, it will be mapped as 1, etc.



**Figure 9 - Mapping Function**

## 5.4.2 Input/Output Sequences for the Network

In this function the length of a sequence is defined at 100 notes/chords, what this ultimately will result in is when the network is instructed to predict the next occurring notes, it will utilize the memory cell of the LSTM network to generalize based on the previous 100 notes in order to formulate an output which is structured on the sequences of the dataset and follows certain pre-existing accords.

The way in which this works is by calculating a formula that determines which note should be output and when. Once a sequence of notes goes through this function an output can be calculated, and this output is based on the first note/chord that comes after the aforementioned input sequence and its set of notes.

```
# create input sequences and the corresponding outputs
for i in range(0, len(notes) - sequence_length, 1):
    sequence_in = notes[i:i + sequence_length]
    sequence_out = notes[i + sequence_length]
    network_input.append([note_to_int[char] for char in sequence_in])
    network_output.append(note_to_int[sequence_out])
```

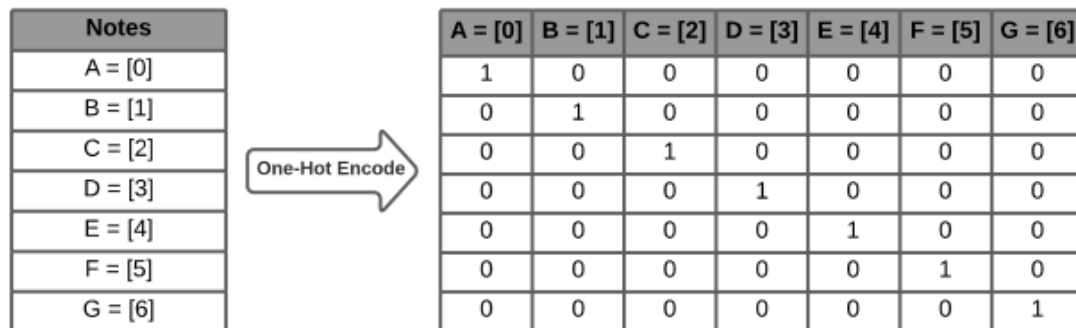
**Figure 10 - Preparation of the input and output sequences used by the network**

The final preparations which occur that ensure the data is ready to be fed into the network are normalization and one-hot encoding. The purpose of normalization is to change the values of numeric columns within the dataset to a common scale. The crux of this principle is to not distort the differences in the ranges of values, otherwise biases may occur which aren't indicative of the actual dataset and the values it holds.

In this instance, normalization occurs by dividing the network's input which is the sequence of notes, by the amount of pitch names that have occurred in said sequence of note. The output of the function is then one-hot encoded, this is to ensure that the algorithm does not assume that higher numbers are more important.

In the previous section when the mapping function was created, the string-based categorical data was converted to integer-based numerical data, and due to their being 7 notes in a key (thus being A, B, C, D, E, F, G), the mapping function would've output values ranging from 0 to 6.

Without one-hot encoding this data the algorithm could potentially assume that the larger numbers are more important which isn't accurate, thus highlighting how critical the place of one-hot encoding is in this implementation. Below is an example of how one-hot encoding would work in the context of this implementation.



**Figure 11 - One-hot Encoding**

## 5.5 Network Architecture & Training

In this section the training process will be described as well as the decisions that were made along with their justification, although prior to actual training the model architecture needs to be defined and this outlines the parameters of the neural network itself. This process of defining the model architecture highlights the major steps that are carried out in the transformation of raw data into training datasets which results in an output that enables the prediction phase of the implementation to have decision making characteristics.

There are four different types of layers that are used within the model which comprise the entire system. Thus being, LSTM layers, Dropout Layers, Dense Layers, and the Activation Layer. Between the Dense Layers and the final Output Layer sits the Sigmoid Activation Function which is where values are converted to 0 and 1 and a decision is made to either allow no flow or complete flow of information.

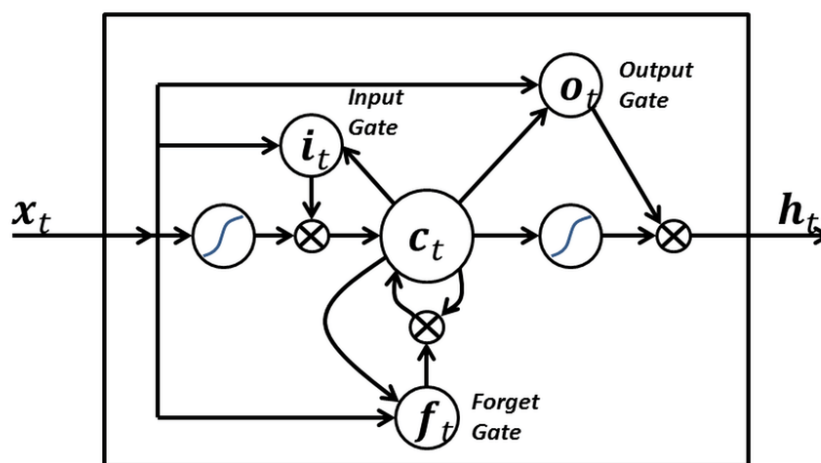


Figure 12 - LSTM Network Architecture - (Zaytar, 2016)

### 5.5.1 Network Layers

LSTM Layers consist of a similar control flow system as RNN's, as it propagates forward, information is passed on as a result of data which has been processed. The key differences between the two are the operations which occur within the LSTM cells, such

as how the memory cell retains important information and forgets data that is no longer relevant through utilization of the forget gate, preventing biases from occurring.

Dropout Layers are used as a regularization technique in which input units are randomly set to 0 with a frequency of rate at each step during training. This is done with the intention of preventing overfitting.

Dense Layers are used for changing the dimensions of the vector by utilizing a deep connection to all of its previous layers in the sense that each neuron in the dense layer continuously receives input from all of the previous layers. The implementation of a dense layer performs the operation:  $output = activation(dot(input, kernel) + bias)$

The Activation Layer is responsible for determining which activation function will be used within the network to calculate the output of a node. The activation function itself serves as a mathematical “gate” in between the input feeding the current neuron and its output which travels to the next layer.

### 5.5.2 Network Specification

Each of the four layers described above functions in a specific way and to a certain extent, this extent is determined by the parameter which follows the call for the layer itself. And regarding the LSTM, Dense and Activation Layers this parameter corresponds to how many nodes the specific layer should have. Whereas regarding the Dropout Layer the parameter represents what percentage of the neurons should be randomly selected and have their weights set to zero for that iteration.

The first layer contains a unique parameter which is referred to as *input\_shape*. This parameter is essential in specifying the shape of the data that will be input to the network because what flows between layers are tensors, and tensors can otherwise be seen as matrices with shapes. The input layer is more accurately a tensor, and it serves as the starting point which is sent to the first hidden layer. The tensor must share the same shape as the training data to prevent catastrophe and this is what makes this parameter so essential.

The output of the network needs to map directly to the classes which in this instance is the notes in the train set, and this is ensured by confirming that the last layer within the network architecture contains a finite number of nodes that is equal to the different number of outputs that exist within the system.

### 5.5.3 Defining Model Parameters

```
def create_network(network_input, n_vocab):  
  
    model = Sequential()  
    model.add(LSTM(  
        512,  
        input_shape=(network_input.shape[1], network_input.shape[2]),  
        recurrent_dropout=0.3,  
        return_sequences=True  
    ))  
    model.add(LSTM(512, return_sequences=True, recurrent_dropout=0.3,))  
    model.add(LSTM(512))  
    model.add(BatchNorm())  
    model.add(Dropout(0.3))  
    model.add(Dense(256))  
    model.add(Activation('relu'))  
    model.add(BatchNorm())  
    model.add(Dropout(0.3))  
    model.add(Dense(n_vocab))  
    model.add(Activation('softmax'))  
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop')  
  
    return model
```

**Figure 13 - Architecture of the network used for training**

The network in this implementation will consist of three fully connected LSTM layers with 512 neurons each. And as the output should appear in the format of a sequence, *return\_sequences=true* is specified to prevent the data from being output in the form of a vector or matrix.

Followed by three Dropout layers with a value of 0.3, meaning that 30% of all neurons will have their weights set to zero at random for a single iteration at a time, the intention of this is to prevent the network from overfitting by forcing it to learn more robust features from the dataset. Although Dropout doubles the number of iterations required to

converge, it reduces the training time required for each epoch, making the training process more efficient.

This will be followed by two Dense layers with a value of 256 which comprises the number of connections in the hidden layer which grant access to all of the previous layers and the neurons which it continuously receives input from.

Finally, the network architecture ends with two activation functions, Relu in the hidden layer, and Softmax in the last layer. The Rectified Linear Activation Function, otherwise known as Relu, is used within the hidden layers and is the activation function of choice because it gives good training and validation accuracy due to the way it handles data sparsity and how it solves the issue of gradient-vanishing. It also makes optimization easier because when the function is active its derivative is set to 1, simultaneously the second derivative of the function is 0 everywhere else in the network.

Softmax is used in the final layer to ensure that the sum of the components of the output sequence is equal to 1, which in turn implies what the probability is for the occurrence of each class, which will contribute toward the prediction process when the model generated from this network is fitted to the prediction environment where the final output in the format of a MIDI file is generated.

The loss for each iteration of training needs to be calculated, and an appropriate approach to this problem would be categorical cross entropy which is a method that utilizes the scalar value in the model output, the corresponding target value, and the output size which is the number of scalar values in the model output.

When trying to distinguish discrete probability distributions from each other, this loss function is a good measure in doing so. The reason being that classification tasks can belong to a particular category with a probability of 1, and to other categories with a probability of zero. And in this instance each output only belongs to a single class and there are several classes which are being utilized.



$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

**Figure 14 - Loss = Categorical Cross Entropy**

The final model parameter that is defined is the optimization method used within the network. RMSprop was chosen because it utilizes an adaptive learning rate which changes over time instead of specifying the learning rate as a hyperparameter which cannot be changed unless training is halted.

$$\begin{aligned} v_{dw} &= \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2 \\ v_{db} &= \beta \cdot v_{db} + (1 - \beta) \cdot db^2 \\ W &= W - \alpha \cdot \frac{dw}{\sqrt{v_{dw} + \epsilon}} \\ b &= b - \alpha \cdot \frac{db}{\sqrt{v_{db} + \epsilon}} \end{aligned}$$

**Figure 15 - RMSprop Update Rule**

During training the process can accelerate and gather momentum, resulting in learning rate schedules which need to change the parameter to help converge the optimization process, otherwise overfitting may occur. By making use of an adaptive learning rate the pressure of specifying the correct learning rate or defining an appropriate learning schedule is alleviated and adjustments can be made automatically based on the loss incurred.

This gradient-based optimization technique makes use of a moving average of squared gradients to normalize the gradient. “This normalization balances the step size (momentum), decreasing the step for large gradients to avoid exploding and increasing the step for small gradients to avoid vanishing.” - (Sanghvirajit, 2020)

### 5.5.4 Fitting the Model

With the network architecture determined and the model parameters defined, it is time to establish training parameters by fitting the model and generated an outcome that can be fit to the neural network in the next stage of development which is also the final stage in which the network will make a prediction based on a generalization from this current training phase.

Before initializing training, a few parameters are defined, beginning with importing the ModelCheckpoint package from Keras which allows the weights from the trained modeled to be saved based on a checkpoint like system. In this instance the model checkpoints will be saved with the epoch number and the validation loss in the filename, this is essential because it allows for the identification of the most successful weight that was trained. The reason it needs to be identified is because during the next phase of the implementation where the prediction will occur, instead of retraining the network, these saved model weights will be loaded.

```
def train(model, network_input, network_output):  
  
    filepath = "weights-improvement-{epoch:02d}-{loss:.4f}-bigger.hdf5"  
    checkpoint = ModelCheckpoint(  
        filepath,  
        monitor='loss',  
        verbose=0,  
        save_best_only=True,  
        mode='min'  
    )  
    callbacks_list = [checkpoint]  
  
    model.fit(network_input, network_output, epochs=200, batch_size=128, ca  
llbacks=callbacks_list)  
  
if __name__ == '__main__':  
    train_network()
```

**Figure 16 - Training the Model**

Keras provides a *model.fit()* function that is used to train neural networks, and the argument consists of several parameters which determine what data is being trained, how much of it, and for how long.

The earlier function which was created to determine the input and output sequences for the neural network satisfies the first and second parameter requirements for the *model.fit()* argument. The first making use of *network\_input* and the second making use of *network\_output*.

The first version of the implementation began with 100 epochs, otherwise known as iterations of training, but the general findings were that it wasn't enough iterations for the network to successfully generalize and as a result the network was overfitting. As a result, in the final version of the implementation 200 epochs were used for training with a batch size of 128 samples (the samples being the training data fed to the network). As each of these batches is propagated through the network it is used to estimate the error gradient by influencing the learning dynamics of the algorithm.

The batch size specified is larger than the entire dataset meaning that it covers every training sample in every iteration, resulting in the iteration and epoch values being equivalent. Using small batch sizes can result in a less accurate estimate of the gradient being calculated.

Lastly, the callback function was used as an object to perform actions at various stages of training, and in doing so it was assigned to *ModelCheckpoints* which resulted in having the model weights saved locally to disk at the end of every epoch.

## 5.6 Prediction & Generating the Final Output

In this separate environment is where the final output for the implementation will be generated in the form of a MIDI file which can be interpreted through the MuseScore external software that will allow for translation into sheet music so that the structure of the musical piece can be evaluated.

The same functions that were developed in the previous environment will be reused to prepare the data and setup the model architecture with the key difference being that instead of retraining on the same network, the pretrained model weights will be imported to the network. It is with this pretrained model that notes will be generated from.

To repeat, using the functions defined in Pipeline Development & Feature Extraction, the same preparation of data will occur to transform the data into the same state as before to make it compatible with the neural network.

The exact same LSTM based neural network will also be used, originally specified in Defining Model Parameters, consisting of the initially mentioned four different types of layers which comprise the entire system. Thus being, LSTM layers, Dropout Layers, Dense Layers, and the Activation Layer. Each maintaining its original parameter values as specified previously.

### 5.6.1 Reverse Mapping Function

In this section the literal reverse will be implemented to that which was created in the Mapping Function section, in the previously defined function the notes present in the dataset were converted from string-based categorical data to integer-based numerical data in preparation for the neural network because training would occur and the most optimal approach to training is with numerical data.

```
### MAP FUNCTION ###  
# create a dictionary to map integers to notes  
int_to_note = dict((number, note) for number, note in enumerate(pitchnames))
```

**Figure 17 - Mapping Integer-based Numerical to String-based Categorical**

Now that the pretrained model weights are being imported and no training is being conducted in this second phase of development in which the final output will be generated, the previously transformed data needs to be remapped back to its initial state which was string-based categorical data so that the notes are in their standard notation and formatting. In doing so the output of the network is being decoded and the list of note sequences is provided for note generation to begin.

### 5.6.2 Generating Notes

The first step in generating notes is declaring a *random.randint* function to initialize within the numpy array that contains the sequences of notes in order to pick a random index within the list as a starting point with the intention of being able to start at a new point each time the generation code is run, therefore producing a unique output on the same dataset.

To give the network as much opportunity as possible to generate a pleasing melody the decision was made to generate 500 notes in the output that'll eventually be written to a MIDI file because 500 notes approximately translate to over two minutes' worth of music, which should be ample time for the network to generalize sufficiently.

```
# generate 500 notes
for note_index in range(500):
    prediction_input = np.reshape(pattern, (1, len(pattern), 1))
    prediction_input = prediction_input / float(n_vocab)

    prediction = model.predict(prediction_input, verbose=0)

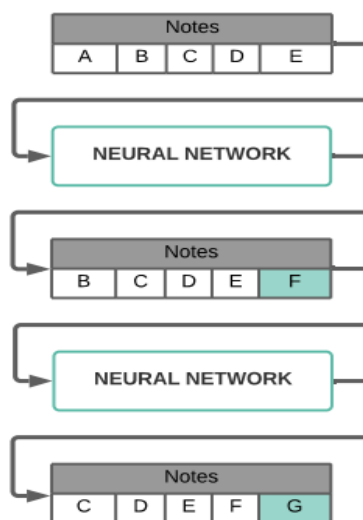
    index = np.argmax(prediction)
    result = int_to_note[index]
    prediction_output.append(result)

    pattern.append(index)
    pattern = pattern[1:len(pattern)]
```

**Figure 18 - Generating notes from the training set**

The previously defined input sequences must now be input to the network for each time a note needs to be generated.

This would mean that the aforementioned starting index which is where the 500 notes will begin generation requires the very first input sequence of notes. Each subsequent note to follow that will make up the musical piece of 500 notes which will require an input sequence per note. The way the network will generalize is by removing the first note from the input sequences, and at the end of each sequence the output of the previous iteration will have its note placed at the end of the sequence.



**Figure 19 - Note Generation from the Input Sequences**

The diagram above explains the process that was mentioned in the previous paragraph, in which the first note from the input sequence will be removed after the first iteration, and the output of the previous iteration is appended to the end of the next sequence.

This is the process of how notes will be selected, but the calculation that actually determines the most likely prediction is based on an extraction of the index with the highest value, using the *argmax()* function provided by Numpy, it's made possible to calculate the indices of classes within the array, therefore figuring out which is the most occurring note in a particular sequence.

The value obtained from the most occurring note in a specific sequence is then removed from said sequence and placed at the end of the following iteration, creating an unending supply of notes which are generalized from the dataset. Ultimately, the output prediction forms a mapping between the LSTM network and the note classes being input to it, and the value of the most occurring note in a specific sequence corresponds to the probability that it may be the next occurring note in the output sequence.

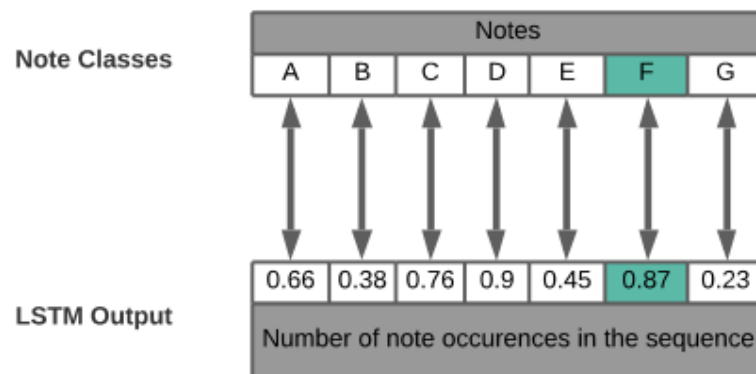


Figure 20 - E.g. How notes are chosen based on frequency of occurrence

### 5.6.3 Write Notes to MIDI file

Now that the network has generalized in which order the notes should occur, the final task is writing said notes to a MIDI file for evaluation. The first step in this task is collecting all of the various outputs from the LSTM network and compiling them into a single array. Seeing as the notes were encoded in preparation for the neural network, they need to first be decoded and placed into an array of note/chord objects, which is the format accepted by Music21 and this software will be responsible for the creation of the final MIDI file.

In the process of encoding the dataset which was completed in the section Dataset Preparation, the notes and chords were distinguished from each other with the intention of making note/chord identification simplified for the network, but now the literal reverse of this function needs to be implemented so that the output being decoded can have both its notes and chords distinguished from one another.

Previously when encoding, the individual notes within a chord were separated by the punctuation mark known as a full stop, e.g., “.”

```
# create note and chord objects based on the values generated by the model
for pattern in prediction_output:
    # pattern is a chord
    if ('.' in pattern) or pattern.isdigit():
        notes_in_chord = pattern.split('.')
        notes = []
        for current_note in notes_in_chord:
            new_note = note.Note(int(current_note))
            new_note.storedInstrument = instrument.Piano()
            notes.append(new_note)
        new_chord = chord.Chord(notes)
        new_chord.offset = offset
        output_notes.append(new_chord)
    # pattern is a note
    else:
        new_note = note.Note(pattern)
        new_note.offset = offset
        new_note.storedInstrument = instrument.Piano()
        output_notes.append(new_note)
```

**Figure 21 - Looping through the generated sequences to separate note/chord objects into stream objects**



As a result, whilst reversing this function the chord object needs to be broken down into an array of notes. Following this step, by looping through the notes which appear in their string format, a note object can be created for each occurring note as this is the format recognized by Music21. From the basis of the note objects which have now been created for every note occurring in a chord, they can now be inserted into a chord object.

The same process is repeated for the identification of notes in their string formats with the intention of transforming them into note objects compatible with Music21, the only difference being that notes are identified by their pitch which is appended to the pattern (list of notes).

The note offset is the duration that a note is held for, and the value is set at 0.5 to prevent notes from stacking against each other, this is done with the intention of creating a coherent sound, and the value 0.5 was previously identified to be the average offset between each note in the dataset.

```
midi_stream = stream.Stream(output_notes)

midi_stream.write('midi', fp='project_output.mid')
```

**Figure 22 - Notes/Chords within stream object are written to an output in MIDI format**

The very final task is appending the note and chord objects created to a list which is the accepted parameter by Music21 for converting the sequence of generated notes into a stream object.

The Music21 package includes a write function that allows for the contents of a stream object to be written to a file, and with the execution of that function, once a file name is specified the final output is generated and the MIDI file is produced.

## 6 Evaluation & Results

### 6.1 Approach to Evaluation of the Algorithm

|                         | <b>Li-Chia Yang et al. (2018)</b> | <b>Nabil Hewahi et al. (2019)</b> | <b>Gaëtan Hadjeres et al. (2016)</b> | <b>Adam Roberts et al. (2016)</b> | <b>Christian J. Walder et al. (2016)</b> | <b>This thesis</b> |
|-------------------------|-----------------------------------|-----------------------------------|--------------------------------------|-----------------------------------|--|--------------------|
| Model Type              | CNN                               | RNN                               | RNN                                  | RNN                               | RNN                                      | RNN                |
| Accepted Data Type      | MIDI                              | MIDI                              | MIDI                                 | MIDI                              | MIDI                                     | MIDI               |
| Genre Specificity       | N/A                               | N/A                               | Bach Chorale                         | N/A                               | N/A                                      | N/A                |
| Follow a Priming Melody | ✓                                 | ✗                                 | ✗                                    | ✓                                 | ✓  | ✗                  |
| Follow a Chord Sequence | ✓                                 | ✓                                 | ✗                                    | ✗                                 | ✗  | ✓                  |
| Versatile Conditions    | ✓                                 | ✓                                 | ✗                                    | ✗                                 | ✓  | ✓                  |
| Open-Source Code        | ✓                                 | ✗                                 | ✗                                    | ✓                                 | ✗  | ✓                  |

**Table 7 - Evaluation of the Algorithm**

The above table demonstrates a list of algorithms that were developed for the purpose of generative music generation, but each has a unique approach to tackling the problem at hand.

Various works achieved by others have contributed to the developments that have been made in this thesis and as a result the key components of each algorithm have been compared and contrasted above to offer insights into how those particular algorithms function and what similarities are shared between them.

The implementation of this project took major inspiration from three separate approaches to build the foundation of this project, from which conclusions could be drawn and evaluation methods established to assess the final output.

The implementations created by Nabil Hewahi et al. (2019), Christian J. Walder et al. (2016), and Sigurður Skúli et al. (2017) are what played a large role in the influence of how this project was developed and the steps taken to reach a final conclusion in terms

of the pipeline development considerations. Further influences came from Li-Chia Yang et al. (2018) in terms of how the neural network was structured and evaluation methodologies.

The first three mentioned works all made use of RNN's to implement the LSTM approach to handling sequence-based data, the primary differences being in how the dataset being input was converted to a format compatible with the LSTM network.

Obtaining the note/chord properties from the dataset in this project was handled by Music21 which is an open-source musicology-based toolkit that is adept at processing the tonal properties of symbolic music such as what is contained within MIDI files, and this is the key difference to what was achieved by Nabil Hewahi et al. (2019), because in their implementation they made use of their own custom MIDI file encoding to their own format to maximise the data manipulation potential.

The custom encoding method implemented by Nabil Hewahi et al. (2019) did not make a distinction between notes and chords, whereas that is the exact purpose of Music21. Instead, the implementation of a sliding window was created by Nabil with a basis focused on time intervals. Within these intervals, notes from the dataset being parsed through the sliding window were captured. This method of capturing notes increased the complexity of their model and in this sense made it superior to the usage of Music21 which was evident in the work of Sigurður Skúli et al. (2017).

The approach to the development of the LSTM network itself was shared commonly between Nabil Hewahi et al. (2019), Christian J. Walder et al. (2016), and Sigurður Skúli et al. (2017). All of which had a basis on predicting the next note to occur based on the previous note sequence. Differences occurred in the structure of the network itself, such as Nabil using 2-4 LSTM Layers, and Sigurður using 3 LSTM Layers. Both presented findings which stated that fewer LSTM Layers with an increased number of neurons on each layer was the most optimal path to generate a satisfying result.

After the evaluation of their findings, a balance was found between the two for the implementation of this thesis, consisting of three LSTM Layers, three Dropout Layers, and two Dense Layers.

## 6.2 Approach to Subjective Evaluation

As discussed previously in the literature review on evaluation methods for generative music creation, two primary branches of evaluation methodologies were identified, thus being summative/subjective testing, and formative/objective testing.

Both forms of evaluation have their lists of pros and cons, beginning with the difficulties of developing a listening based experiment for subjective testing which is rigorous enough to appropriately evaluate the music with enough participants to out rule a general bias, which typically suffers due to a lack of resources. And ending with objective testing which takes the user and any of their potential biases out of the equation, and strictly performs experiments based on the structural and tonal properties of the music piece in question, but this form of evaluation often lacks relevance because the metrics can only cover so many avenues, and when dealing with music and the emotion/creativity that is often associated with it, the obvious choice must be a listening based experiment, but that doesn't entirely rule out the need for some form of objective evaluation too.

### 6.2.1 Listening Based Experiment

With the evaluation methodology decided, a listening based experiment was designed to evaluate the aesthetic quality of the generation result. The user study was conducted with 10 participants, all of which were either amateur musicians and/or had a basic understanding of music theory and the concepts associated with it. The people chosen to partake in this study could therefore be considered as people with musical backgrounds.

From the prediction environment 20 MIDI outputs were generated, each with a playback duration of approximately 2 minutes, and each MIDI file from the training dataset consisted of a 4-bar structure within a 4/4 time signature, meaning that there are 4 quarter beats in each measure, with a typical stepsize of 8 notes per bar, and a note offset of 0.5, meaning that the time taken from the start to end of an individual note is typically 0.5 seconds.

Of these 20 MIDI file outputs, a single file was randomly selected for the users to evaluate, and from this single file a 30 second excerpt was produced and this 30 second musical piece is exactly what they evaluated during the listening based experiment.

On average the musical structure of the training dataset is reflected throughout the prediction output results, but there are some interesting exceptions where the network generalized in unique ways, are described below.

### 6.2.2 Sheet Music of the excerpt that was Evaluated

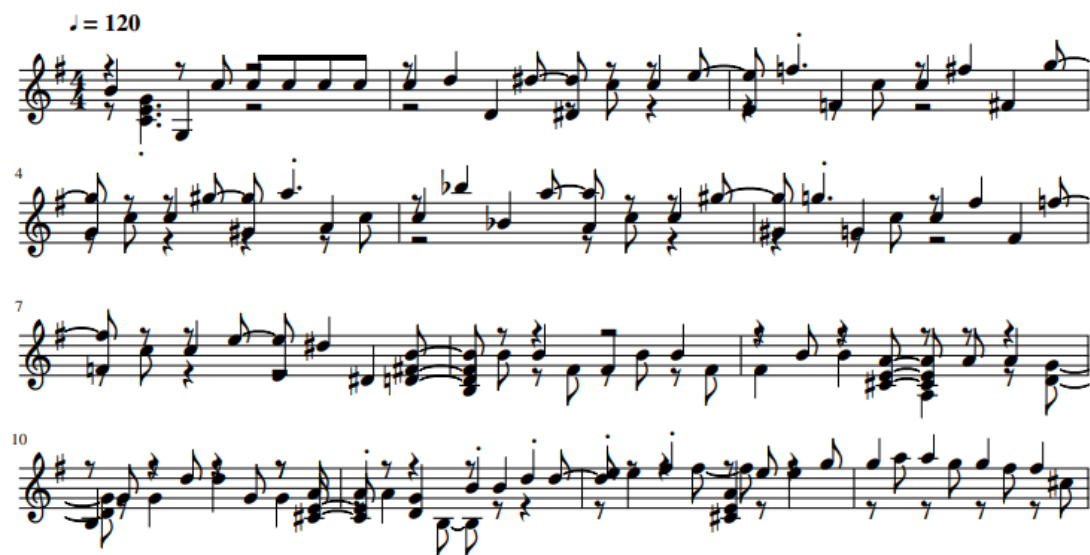


Figure 23 - Sheet music of file 'project\_output\_3.MID'

The above excerpt from the MIDI output that was evaluated contains a sufficient amount of musically accurate structure which can be broken down into a few categories. Firstly, the number of single notes present in the entire MIDI file is 120, the 4/4 time signature means that there are 4 quarter beats in each measure, and each bar typically consists of 8 notes. The symbol found on the far left of each measure is a treble clef, and this symbol primarily notates musical notes above middle C.

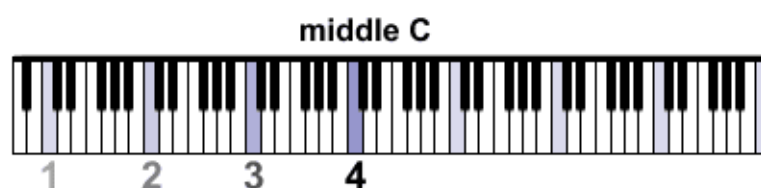


Figure 24 - Identification of Middle C on the full piano keyboard

This identification of musical notes occurring above middle C means that most of the notes featured within this musical excerpt appear on the higher end of the piano, thus resulting in sharper sounds instead of the lower monotone notes associated with the section of the piano below middle C.

### **6.2.3 User-based Study Criteria & Results**

The listening based experiment was conducted through a user study consisting of 10 participants that each had a musical background to some extent. Each participant of the study was sent the 30 second duration audio excerpt through the internet and asked to listen to the piece with headphones on and in a controlled environment.

Whilst listening to the musical piece, participants of the study were asked to evaluate it based on three criteria:

- **How pleasing did the audio sound?**
- **How realistic did the audio sound?**
- **How interesting did the audio sound?**

Regarding how pleasing the audio sounds, is in referral to the aesthetic quality of the music. Regarding how realistic the audio sounds, is in referral to whether or not they could determine if the music was generated by a computer or composed by a human. And regarding how interesting the audio sounds is in referral to whether or not the tonal properties of the audio piece piqued their interest.

With all of these criteria in mind, the participants were asked to score each condition on a scale from 1 to 5 (1 being the lowest possible score and 5 being the highest possible score).

The results of the study are as follows:

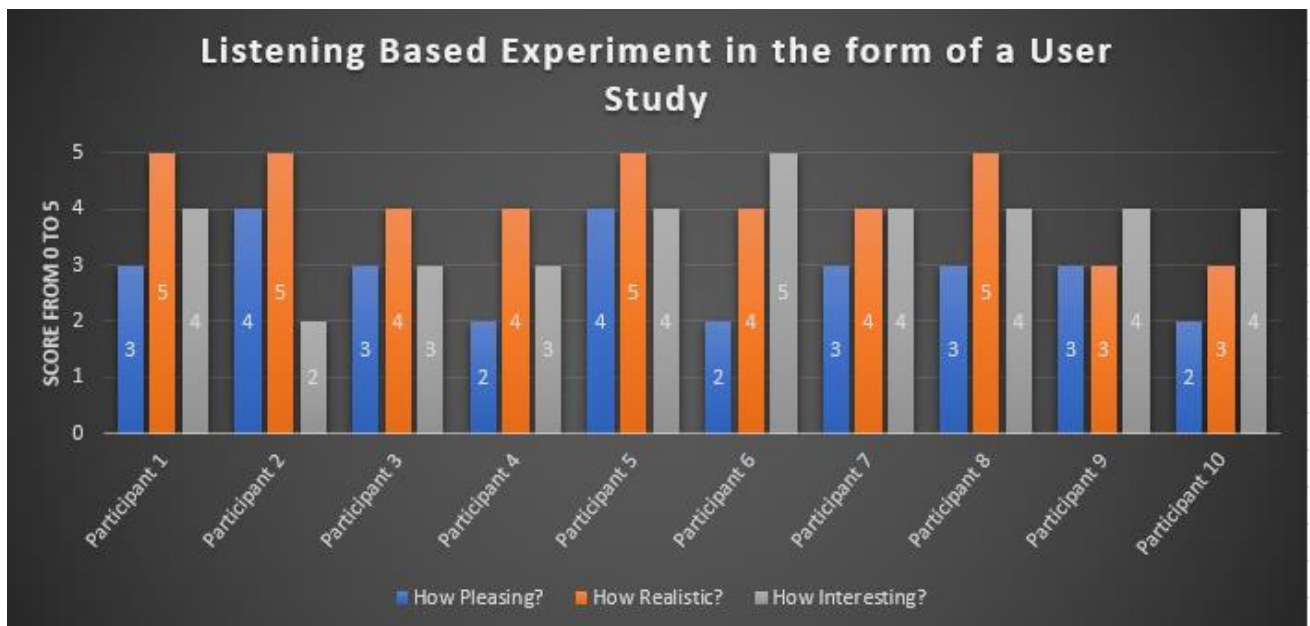


Figure 25 - Results of the user-based study

#### 6.2.4 Mean Value of Results

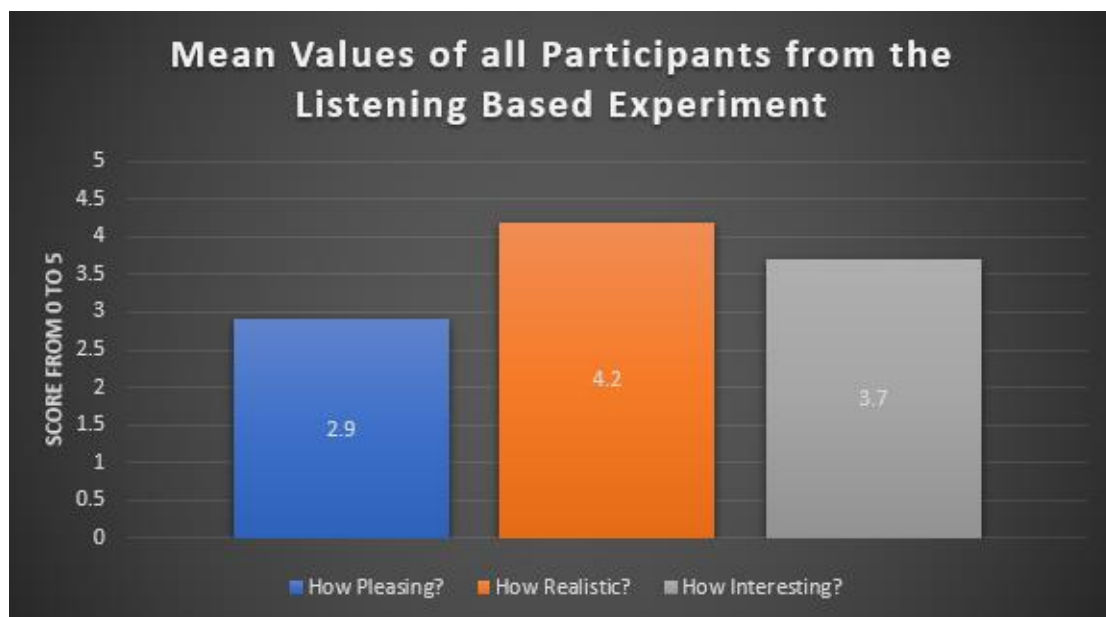


Figure 26 - Mean value of the results obtained from the user study

The average score for how pleasing the participants of the experiment found the audio piece was 2.9, the average score for how realistic they believed the audio piece to be was 4.2, and the average score for how interesting they found the audio piece is 3.7.

These results are extremely interesting and reflect the capability of the neural network and how it was able to generalize based on the training dataset sufficiently to create a realistic and interesting sounding piece of music.

The lowest value being 2.9 was a reflection of how pleasing the participants of the study found the audio piece to sound and their findings are somewhat indicative of the approach taken toward note generation in the final output. The reason being that whilst note and chord classes were made for interpretation by the neural network, varying rest classes weren't, and as a result a finite predetermined value of 0.5 was applied as the note offset to each and every note/chord present in the MIDI file outputs, meaning that the note duration is exactly 0.5 seconds.

In music rests and their durations are essential to the way in which a musical piece sounds and is portrayed, a lack of varying rests between notes can greatly hinder the impact that a musical piece is able to have, and in this instance may be the reason that the participants of the study on average did not find the audio piece to be particularly aesthetically pleasing. But regarding how realistic and interesting the audio piece sounds received above average scores, meaning that the network was successfully able to generalize on the training set to an extent that the average musically inclined user could not distinguish it from a musical piece composed by a human, and it also sounded interesting enough to pique their curiosity and fascination.



### 6.2.5 Participant Feedback

It was not a requirement for participants of the listening based experiment to give any other feedback beside the three specified criteria, but some opted to give their verbal feedback on what they thought about the musical piece that they evaluated and how it made them feel.

The results are as follows:

|                      | <b>How pleasing did it sound?</b>   | <b>How realistic did it sound?</b>   | <b>How interesting did it sound?</b>   |
|----------------------|---|--|--|
| <b>Participant 1</b> | <u>Score = 3</u><br><br><u>Response:</u><br>“Very high pitch, for me needed to be a bit lower in pitch”   | <u>Score = 5</u><br><br><u>Response:</u><br>“As realistic as any other classical music”                | <u>Score = 4</u><br><br><u>Response:</u><br>“I like classic music so it’s quite interesting that it could be totally produced by a computer, just like other forms of art/ paintings generating huge money nowadays” |
| <b>Participant 2</b> | <u>Score = 4</u><br><br><u>Response:</u><br>“It's rather unsettling for me; I doubt that I would listen to it in my own, however, it sounds like a great piece for a thriller/drama movie, because of this adaptational versatility I will give it 4” | <u>Score = 5</u><br><br><u>Response:</u><br>“I could never tell that it's not a person, therefore – 5” | <u>Score = 2</u><br><br><u>Response:</u><br>“I can't see any uniqueness in this piece and since I said that it could've been a person who played/composed it it's rather mediocre, I would give it 2”                |

|                      |  |  |   |
|----------------------|--|--|---|
| <b>Participant 3</b> | <p><u>Score = 3</u></p> <p><u>Response:</u></p> <p>“The pitch is too high for my ears, but that’s just me, I can’t handle some sounds so not sure if this answer helps”</p>  | <p><u>Score = 4</u></p> <p><u>Response:</u></p> <p>“I can definitely imagine a person playing that but at the same time It felt that it lacked something that usually classical music makes me feel - or I already had prejudices for knowing that it’s not a real person”</p> | <p><u>Score = 3</u></p> <p><u>Response:</u></p> <p>“Fairly interesting, I can definitely imagine a part of that in a B-movie thriller. I mean, it’s not Hans Zimmer but definitely creates a state of suspense”</p>   |
| <b>Participant 4</b> | <p><u>Score = 4</u></p> <p><u>Response:</u></p> <p>“The pitch was a bit high but not so much so that I didn’t enjoy it”</p>  | <p><u>Score = 5</u></p> <p><u>Response:</u></p> <p>“I wouldn’t be able to tell if a computer made this audio or not”</p>   | <p><u>Score = 4</u></p> <p><u>Response:</u></p> <p>“I could imagine this music being part of a film”</p>  |
| <b>Participant 5</b> | <p><u>Score = 2</u></p> <p><u>Response:</u></p> <p>“I enjoy piano and classical music however the sounds were slightly too erratic and unsettling. I wouldn’t personally listen to it through my earphones for example.”</p> | <p><u>Score = 4</u></p> <p><u>Response:</u></p> <p>“It sounds quite realistic to me like an electric keyboard. If this sound was attached to a video of a person playing the keyboard I wouldn’t doubt that the sound is coming from the keyboard.”</p>                        | <p><u>Score = 5</u></p> <p><u>Response:</u></p> <p>“I actually find the sound to be very interesting and narrative in a sense. I can imagine it playing in the background of an old school silent film such as The Oyster Princess. The ups and downs of the sound would complement a dramatic story”</p> |

**Table 8 - Verbal Participant Feedback (Optional)**

### 6.3 Approach to Objective Evaluation

| Dataset MIDI Files     |              |                 |       |       |                |                 |
|------------------------|--------------|-----------------|-------|-------|----------------|-----------------|
| Track Name             | Instrument   | Number of Notes | Tempo | Ticks | Low Tone Notes | High Tone Notes |
| Bicycle-ride.mid       | Bright Piano | 180             | 133   | 384   | 72             | 88              |
| Bicycle-ride-2.mid     | Bright Piano | 402             | 140   | 120   | 67             | 81              |
| Celadon-city.mid       | Bright Piano | 330             | 137   | 120   | 62             | 79              |
| Celadon-city-remix.mid | Bright Piano | 547             | 95    | 180   | 73             | 87              |
| Cerulean-city-3.mid    | Bright Piano | 206             | 132   | 120   | 59             | 78              |

**Table 9 - Objective Evaluation of Dataset**

| Generated MIDI Files |              |                 |       |       |                |                 |
|----------------------|--------------|-----------------|-------|-------|----------------|-----------------|
| Track Name           | Instrument   | Number of Notes | Tempo | Ticks | Low Tone Notes | High Tone Notes |
| Project_output_1     | Bright Piano | 500             | 120   | 180   | 52             | 81              |
| Project_output_2     | Bright Piano | 500             | 120   | 180   | 54             | 91              |
| Project_output_3     | Bright Piano | 536             | 120   | 180   | 55             | 91              |
| Project_output_4     | Bright Piano | 500             | 120   | 180   | 41             | 85              |
| Project_output_5     | Bright Piano | 647             | 120   | 180   | 59             | 105             |

**Table 10 - Objective Evaluation of Generated MIDI**

The dataset consists of 80 samples and the algorithm has generated 20 samples. Due to the sheer number of individual tracks to deal with regarding objective evaluation, the first 5 tracks from both the input dataset and generated output have been used to form the basis of the objective segment of evaluation.

As can be seen from the first table, all 5 tracks have varying parameters to do with the number of notes, tempo, ticks, etc. This was regulated when generating the output of the network and is the reason why the generated MIDI files don't largely vary in number of notes between each track. The purpose behind this was to regulate the duration of each track so that there was ample time for the network to generalise and create a pleasant sound.

The number of notes corresponds to how many notes are present in each track, the tempo refers to the speed or pace of the entire track, the number of ticks refers to a tempo-dependent measurement of musical time. The size of which is defined by the time division of the MIDI track. Low tone notes refer to the number of notes occurring on the lower end of the piano, producing deeper sounds. And high tone notes refer to the number of notes occurring on the opposite side of the piano, being the higher notes, producing a sharp sound.

In the input dataset there is a somewhat equal balance between the number of low tone notes and high tone notes, producing a more melodic and balanced sound. Whereas in the generated MIDI files there is a definite favouritism toward high tone notes, which is as a result of the network forming biases whilst training.

This metric is reflected in the listening based experiment where several of the participants mentioned that the generated MIDI file was relatively high pitched, and this resulted in the lowest overall score from the listening based experiment belonging to the criteria "How pleasing did the audio sound?".

### 6.3.1 Objective Evaluation of a Sample from the Dataset

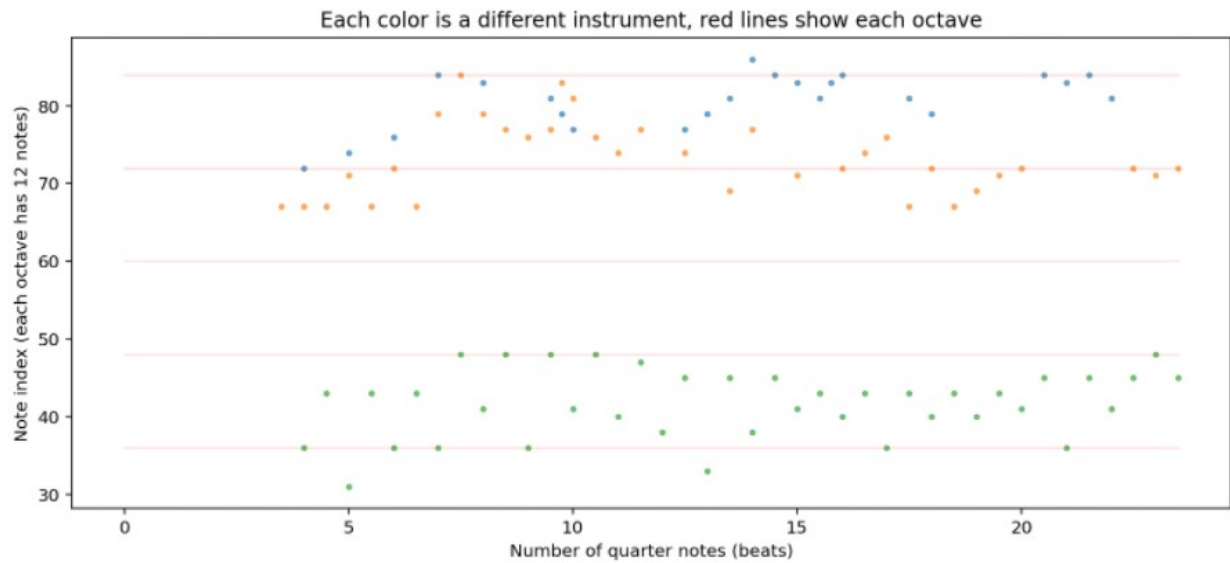


Figure 27 - Dataset Sample Line Plot

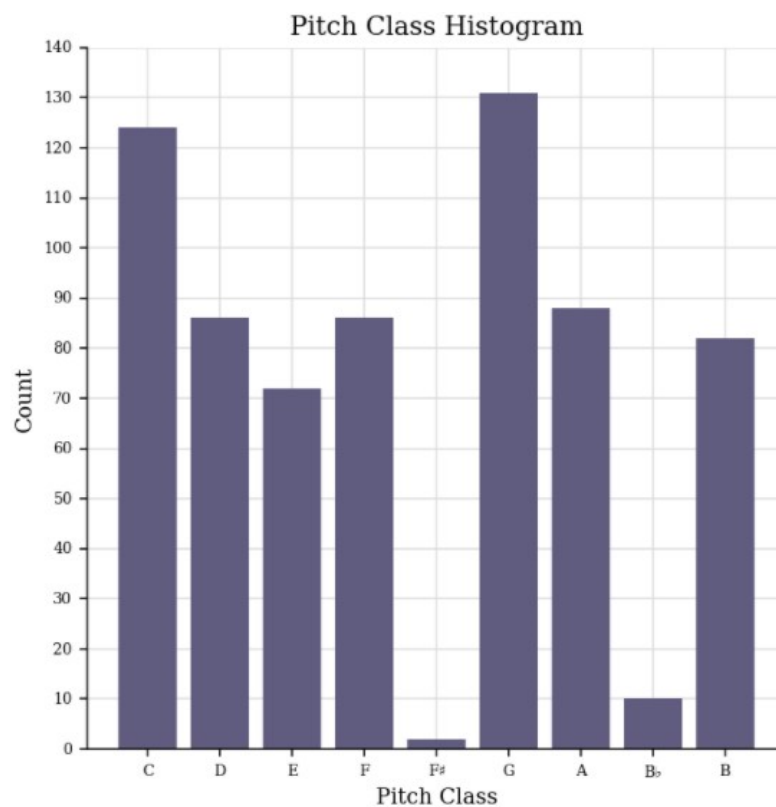
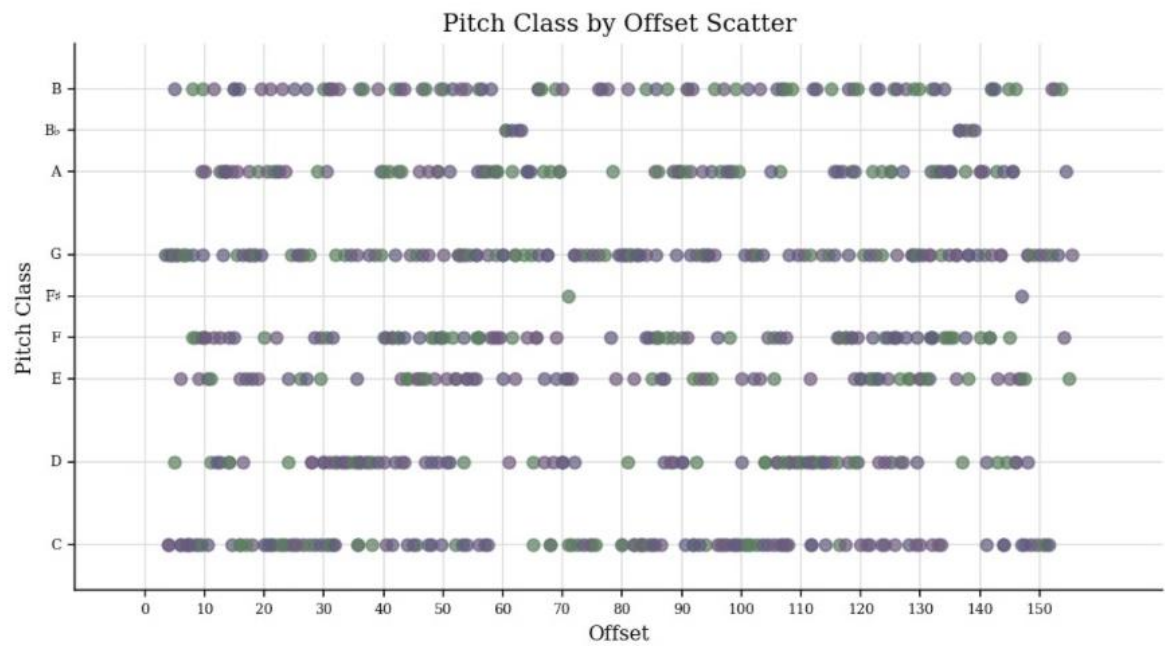


Figure 28 - Dataset Sample Histogram



**Figure 29 - Dataset Sample Scatter Plot**

Music time signature: 4/4  
 Expected music key: C major  
 Music key confidence: 0.8562083997782167  
 Other music key alternatives:  
 G major  
 a minor  
 d minor  
 F major  
 g minor

**Figure 30 - Dataset Sample Composition Parameters**

### 6.3.2 Objective Evaluation of a Sample from the Generated Output

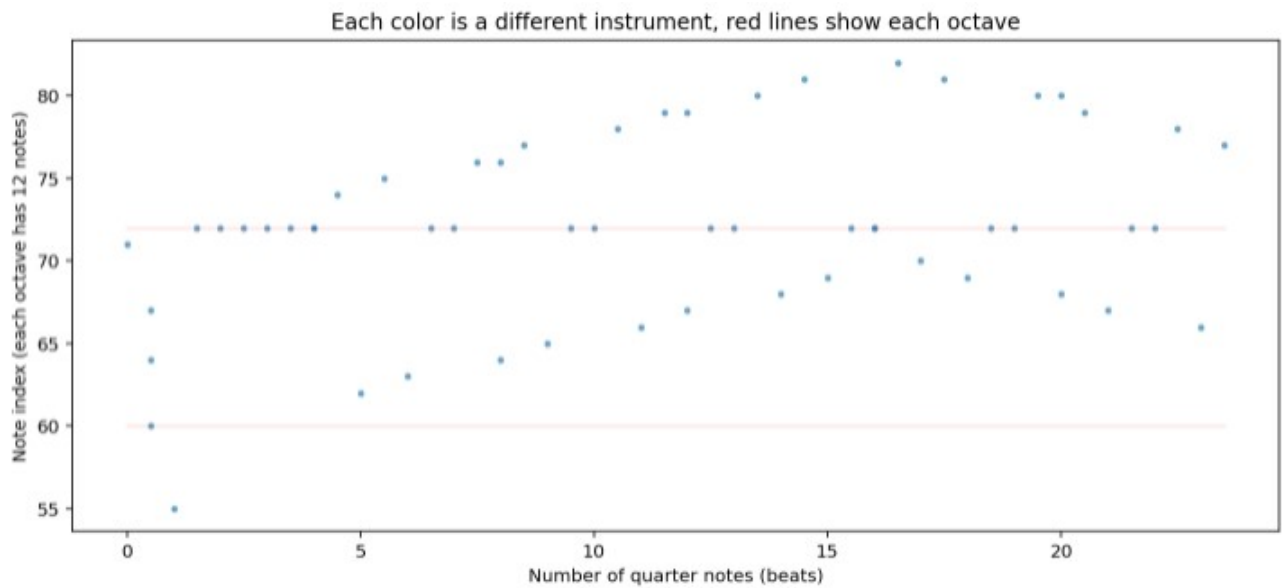


Figure 31 - Generated Output Line Plot

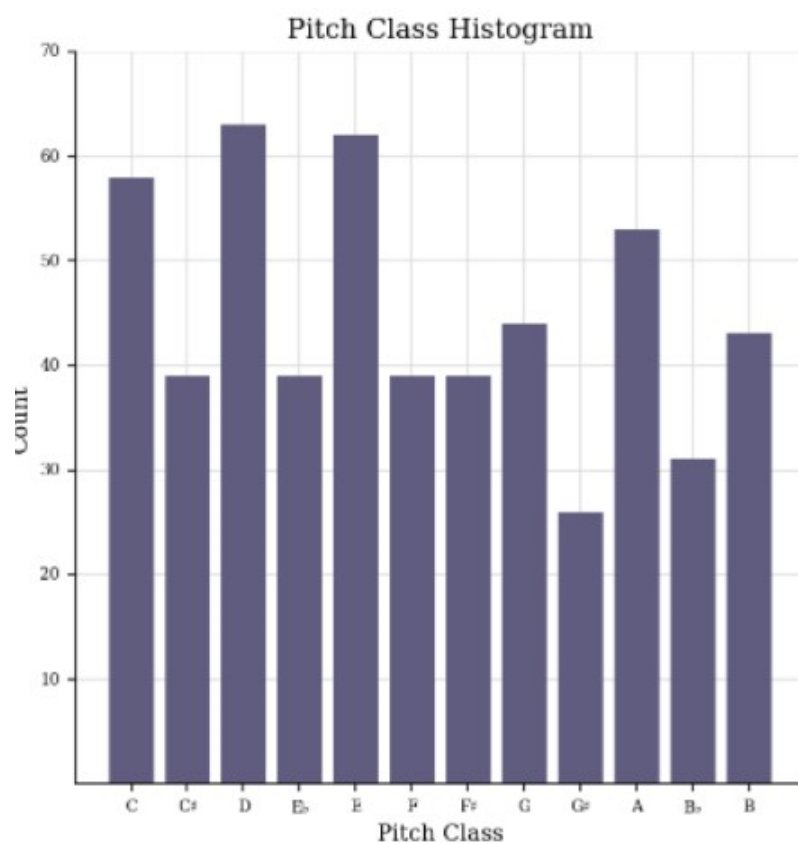


Figure 32 - Generated Output Histogram

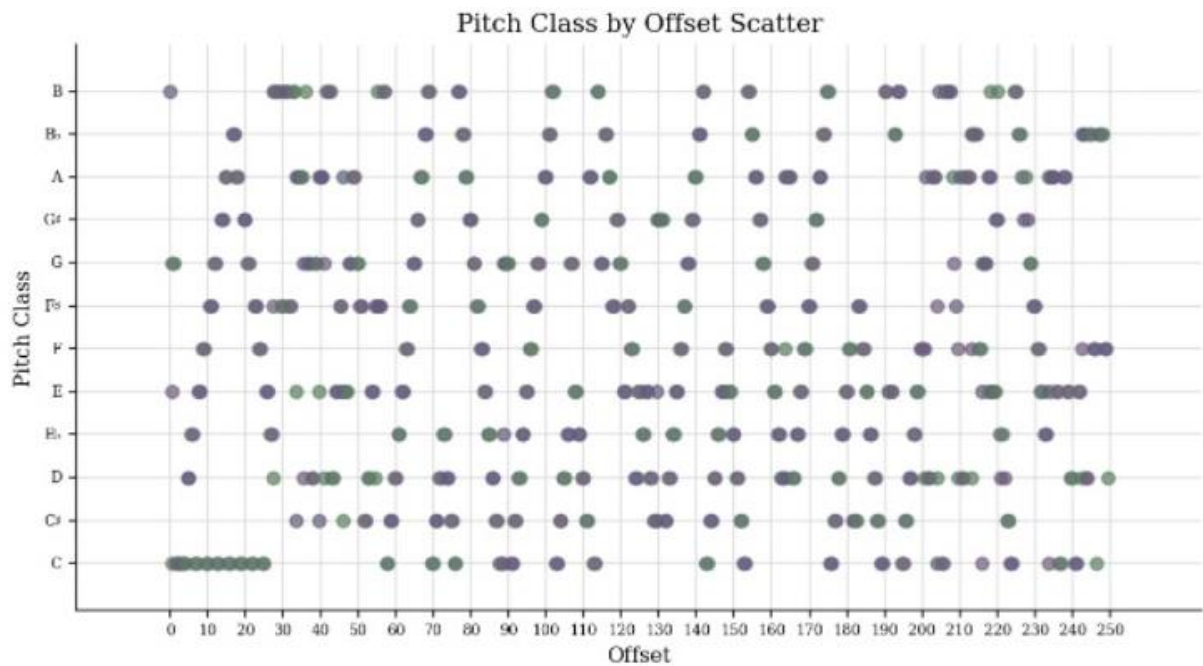


Figure 34 - Generated Output Scatter Plot

Music time signature: 4/4  
 Expected music key: a minor  
 Music key confidence: 0.864878743601705  
 Other music key alternatives:  
 C major  
 G major  
 d minor  
 D major

Figure 33 - Generated Output Composition Parameters



### 6.3.3 Conclusions on Objective Evaluation

Regarding the first parameter of the first graph, the number of instruments present in the MIDI file which is represented by different colors is also representative of instrument layering. For example, in the first sample taken from the dataset several instruments can be seen ranging across the different octaves of the piano, but what's actually happening in this sample is a piano has been layered to have multiple melodies played simultaneously, creating a larger depth of sound.

This did not reflect in the generated output which was not designed to layer instruments. This is something that could potentially be developed for future work by allowing the acceptance of more than one instrument to the network which currently isn't possible.

The pitch class histogram is representative of the range of pitches present in both a sample from the dataset and a sample from the generated output. As can be seen from the diagram, the generated output did an exceptional job at sufficiently generalizing on the entire dataset, by using a wide range of pitches across the entire note range of the piano, instead of forming biases by displaying favoritism on independent note pitches.

The scatter plot is reflective of when certain notes are being played in relation to note duration, which is useful in potentially detecting the change of key signatures. The first scatter plot appears more normal because it reflects an individual MIDI file from the dataset. Whereas the generated output is not a single well formulated musical piece. It is a combination of 80 different samples, used to form a generalization which is then reflected in the final output. As can be seen from the diagram, the scatter plot is particularly erratic for the generated output, meaning that there are many changes in key signatures throughout the track.

The final plot is of some composition parameters pertaining to each MIDI file in question for the objective evaluation. These parameters include metrics such as the time signature which is indicative of how many beats are contained in each measure. As well as the expected music key within the piece.

## 7 Conclusion

### 7.1 Overview

In this thesis a novel implementation of a Long Short-Term Memory based Recurrent Neural Network has been created with the sole intention of generative music composition in the format of a MIDI file.

The implementation took place in two separate environments which were tasked with specific duties. The first environment was responsible for processing the dataset into a state compatible with the LSTM network by mapping the data from categorical to numerical, and then one-hot encoding it before training a model that has an input of two parameters. The first of which being the list of input sequences that were created previously, followed by the list of their respective outputs.

The second environment was responsible for the prediction of notes which began with importing the notes which were mapped and one-hot encoded and then importing the pre-trained model weights from the training environment and performing neural network inference on the data by applying the knowledge obtained from the pre-trained neural network to perform a prediction.

Once the prediction was performed on the sequence of data by use of NumPy's *argmax()* function, a series of notes were generated. The notes and chords generated needed to be decoded and parsed through Music21 to separate the note/chord objects into a stream object which finally wrote the output to a MIDI file for further evaluation.

A listening based experiment was performed in the form of a user study which consisted of 10 participants that all had a musical background to some extent, with the intention of evaluating the generated musical piece in a sufficient and decisive manner. The results of the evaluation proved that the network was successfully able to generalize to the extent that a musical piece was produced that achieved above average scores in all three scoring criteria based on how pleasing, how realistic, and how interesting the music sounded.

## 7.2 Requirement Fulfilment

| <b><u>Functional Requirements:</u></b> |   |   |  |
|--|---|---|--|
| <b><u>Entity:</u></b>                  | <b><u>Requirement:</u></b>  | <b><u>MoSCoW</u><br/><u>Prioritisation:</u></b> | <b><u>Achieved?</u><br/><u>(Y/N)</u></b> |
| System/Environment                     | Create a network that can successfully predict on the training set. | M   | Y  |
| System/Environment                     | Generate notes from the trained network.                            | M   | Y  |
| System/Environment                     | Input a dataset consisting of MIDI files.                           | M   | Y  |
| System/Environment                     | Convert notes/chords into binary format.                            | S   | Y  |
| System/Environment                     | Make use of a LSTM neural network.                                  | S   | Y  |
| System/Environment                     | Evaluate the final output.  | S   | Y  |

**Table 9 - Functional Requirement Fulfilment**

| <b><u>Non-functional Requirements:</u></b> |  |   |  |
|--|--|---|--|
| <b><u>Entity:</u></b>                      | <b><u>Requirement:</u></b>   | <b><u>MoSCoW</u><br/><u>Prioritisation:</u></b> | <b><u>Achieved?</u><br/><u>(Y/N)</u></b> |
| Usability                                  | The algorithm must meet certain criteria that determine the project a success.     | M   | Y  |
| Maintainability                            | The code written for the algorithm must be maintainable, updatable, and efficient. | M   | Y  |
| Required Resources                         | The algorithm itself must be trainable and functional on the system.               | M   | Y  |
| Platform                                   | The algorithm must be compatible with different software versions.                 | S   | Y  |
| Readability                                | The code should be readable and understandable.                                    | S   | Y  |
| System/Environment                         | The trained model weights should be savable.                                       | C   | Y  |
| System/Environment                         | The trained model loss value should be calculated.                                 | C   | Y  |
| Response Time                              | The software should have fast response times whilst in use.                        | C   | Y  |

**Table 10 - Non-functional Requirement Fulfilment**

The functional requirements have been provided in a low-level style to encapsulate the various benchmarks of the implementation aspect of this project. The intention behind this being that due to this project having many minor technical aspects, it could potentially be difficult to identify the relevance in mentioning each individual aspect.

As a result, the core principles behind the implementation have been broken down into 6 categories that encapsulate the technical processes being conducted. And an overall satisfactory result has been achieved because every single functional requirement has been fulfilled to generate the final output which is the musical piece in the format of a MIDI file. And instead of evaluating the training process which lead to the final result, evaluation was strictly conducted on the result itself which is indicative of a successful training process and centers the focus around the actual music that was produced and its musical structure, by assessing the tonal properties of the sound itself.

The non-functional requirements specify criteria that can be used to judge the operation of the system that has been developed, rather than specific behaviors which are outlined in the functional requirements section.

The purpose of this is to enable the most optimal and efficient path to success whilst ensuring the usability and effectiveness of the entire system and the outcomes being achieved. A satisfactory result has been achieved regarding the non-functional requirements as each metric has been satisfied.

## 7.3 Findings and Recommendations

The findings and recommendations of this project can be assessed by addressing the research questions that were previously established in the beginning of this report. The research questions encapsulate the direction taken toward development, and in completing the project key discoveries were made regarding what information has been identified and how it has been applied to the project.

### **1. Does Artificial Intelligence have a place in the domain of music?**

This open-ended question is what opened the gateway for this project to exist. As previously discussed, it is a controversial question in the eyes of many members of the music industry due to fears of becoming obsolete due to the advancement of machine learning.

Modern algorithms at the forefront of generative music composition are incredibly sophisticated with some able to create a host of symphonies that resemble some of the greatest musicians of our time. And although what has been achieved in this project is a novel implementation in comparison to these feats, it is a proof of concept that serves as an indicator of the presence of machine learning in the music industry and how it isn't created as a replacement to musicians.

It is instead created with the intention of a symbiotic relationship between both musician and algorithm to carve a new path forward where a human can provide the creative directive and inspiration as a starting point for an algorithm to develop upon and potentially make new discoveries in the field of music because there are many structural properties surrounding the field which are engulfed in mathematics and this is the factor that allows machine learning to thrive.

**2. Is LSTM the appropriate approach to develop a network that is capable of generative models in music?**

| <b><u>Epoch</u></b> | <b><u>Batch Size</u></b> | <b><u>Steps Per Epoch</u></b> | <b><u>Time Taken</u></b> | <b><u>Loss</u></b> |
|---------------------|--------------------------|-------------------------------|--------------------------|--------------------|
| 10                  | 128                      | 361                           | 79s 217ms                | <b>3.7315</b>      |
| 20                  | 128                      | 361                           | 79s 218ms                | <b>3.1523</b>      |
| 30                  | 128                      | 361                           | 79s 218ms                | <b>2.3852</b>      |
| 40                  | 128                      | 361                           | 79s 218ms                | <b>1.7156</b>      |
| 50                  | 128                      | 361                           | 79s 218ms                | <b>1.0798</b>      |
| 60                  | 128                      | 361                           | 79s 218ms                | <b>0.6419</b>      |
| 70                  | 128                      | 361                           | 79s 218ms                | <b>0.4015</b>      |
| 80                  | 128                      | 361                           | 79s 218ms                | <b>0.2829</b>      |
| 90                  | 128                      | 361                           | 79s 218ms                | <b>0.2123</b>      |
| 100                 | 128                      | 361                           | 79s 218ms                | <b>0.1663</b>      |
| 110                 | 128                      | 361                           | 78s 217ms                | <b>0.1405</b>      |
| 120                 | 128                      | 361                           | 79s 217ms                | <b>0.1280</b>      |
| 130                 | 128                      | 361                           | 79s 217ms                | <b>0.1077</b>      |
| 140                 | 128                      | 361                           | 79s 218ms                | <b>0.0986</b>      |
| 150                 | 128                      | 361                           | 79s 217ms                | <b>0.0889</b>      |
| 160                 | 128                      | 361                           | 79s 218ms                | <b>0.0827</b>      |
| 170                 | 128                      | 361                           | 79s 217ms                | <b>0.0782</b>      |
| 180                 | 128                      | 361                           | 79s 217ms                | <b>0.0750</b>      |
| 190                 | 128                      | 361                           | 78s 217ms                | <b>0.0717</b>      |
| 200                 | 128                      | 361                           | 78s 217ms                | <b>0.0698</b>      |

**Table 11 - Results from Training the Network**

This question was easily addressable before implementation even began due to the research conducted on the different types of neural networks that were appropriate for this implementation.

Long Short-Term Memory, otherwise known as LSTM, belongs to the group of networks referred to as Recurrent Neural Networks and this network specializes in working with sequences of data rather than the typical singular data points the often occur in datasets.

What this ultimately means is that it is adept at handling sequence-based data due to the memory cell that exists within the structure of the network and how it retains a store of information for a certain duration of time which is long enough for the network to perform predictions on the dataset, otherwise referred to as generalizations. As a result of this memory cell the network is also able to determine which data is no longer important and discard it to prevent biases from occurring in training.

The result of this after a substantial amount of training was a loss calculation of 0.0698, what this ultimately means is that the model did a great job at prediction because the loss figure is indicative of how good or bad the model's prediction was on a single example.

Training on the first epoch began with a loss calculation of 4.6985 and on the 200<sup>th</sup> epoch had the aforementioned value of 0.0698, meaning that the training process was a success, and the network was successfully able to generalize on the dataset.

It is with these results that a conclusion regarding the appropriateness of LSTM can be made, and the results of training followed by an overall successful evaluation is proof that the approach to utilize a LSTM based Recurrent Neural Network for the creation of generative music was effective and justifiable.

### **3. Is the MIDI file format the most appropriate communications protocol for obtaining the tonal properties from an audio file?**

Depending on the approach to development and how a developer plans to implement their algorithm, there are several different approaches that way be considered, each of which utilizing different methodologies that require data in different formats.

The research question being addressed in this section is to do with the format of the chosen dataset that was later input to the neural network. This question covers one of the most important aspects of this project because data type plays a large role in how a neural network is able to interpret and process data and it also contributes toward the decisiveness of the final output.

The decision to use the MIDI file format, otherwise known as a Musical Instrument Digital Interface, is based on how the file stores its contents and what precisely it stores. When looking at other audio file formats such as MP3, these formats contain the actual audio file stored within it, meaning that the extraction of tonal properties such as notes, chords, pitches, etc. can be extremely difficult. And in most cases an interpreter would be needed to map the contents of the audio to its structural output, but this path can lead to errors and miscalculations.

The MIDI file format does not contain an audio file within it, instead, within the MIDI files a set of instructions is stored that explain to an interpreter how the audio is produced, and the audio interpreter decodes the instructions within the file and produces a sound.

Regarding this project, it was absolutely essential to obtain these tonal properties pertaining to the dataset and so the obvious choice was the MIDI file format. Once the contents of the dataset were parsed through Music21 and back to Python, the musicology toolkit was able to provide every aspect of the tonal properties such as the notes and chords present, the rests in between, the notes offsets, their relevant pitches, etc. Deeming the MIDI file, the ultimate format to tackle the task.



#### **4. Are summative or formative based experiments the most appropriate for evaluating music?**

Regarding evaluation methods in relation to generative music composition, the literature review explored two main avenues. Thus being, summative/subjective testing, or formative/objective testing.

As previously discussed, both methods of evaluation contain their pros and cons such as a subjective approach to evaluation potentially lacking the resources to conduct a thorough and decisive evaluation in the form of some kind of listening based experiment that provides enough evidence to form a solid conclusion.

Alternatively, the objective based approach to evaluation can lack relevance due to the parameters being measured not being meaningful in the sense that calculating the validation/loss accuracy of a network might indicate the network was slightly overfitting on the data, but what that means in terms of the sound being produced is absolutely meaningless.

Other approaches to objective evaluation can measure criteria such as note occurrences, rhythmic patterns, tonal distance, pitch classes, etc. But these parameters potentially lack the relevance in establishing the quality of the actual sound being produced because music as a whole is a subjective matter that cannot be fit within a set of criteria due to reasons such as the emotion and creativity that are often associated with the sound and how it makes a potential listener feel.

Therefore, leading to the conclusion that the most appropriate way to evaluate the produced MIDI output of this project is by means of a listening based experiment in order to maintain relevance to the actual topic instead of evaluating on a meaningless set of criteria that do not indicate how pleasing the produced audio might actually sound.

## **7.4 Neglected Considerations & Further Development**

Neglected considerations and further development coincide with each other to an extent within this project because some potential developments in the project that could've contributed to a more concise final output but due to time constraints they have been left for further development because although they offer potential improvements, the justification does not offset the time constraints.

### **7.4.1 Increase the size of the Dataset**

The first neglected consideration being the size of the dataset, which ultimately corresponds to the depth of the neural network. The dataset as it is now contains only 80 samples and the neural network contains 3 LSTM layers with 512 neurons in each layer. By increasing the size of the dataset from 80 samples to say 500 samples, it is a significantly larger amount of data for the network to perform predictions based on, therefore potentially allowing a more musically complex and diverse MIDI output to be generated.

With the upscaling of the dataset, the neural network would need to have its depth increased too, potentially with more LSTM layers and more neurons per layer to create a vast network of connections that ultimately lead to a much longer training process, but a more accurate result.

### **7.4.2 Adding classes to handle varying note lengths/durations**

The next improvement that is both a neglected consideration and a means for further development is to add support for varying note lengths, as well as unique offset values between each note, whereas currently it's a static value set at 0.5. The way this problem could be tackled is by adding a new class that handles these rest periods between notes, and also for the handling of note durations. This would contribute to a more creative sound as music is typically structured around varying note durations and rests between notes/chords to add cadence and/or an aspect of thrill/anticipation.

With the addition of these improvements the depth of the network would have to be increased sufficiently, which as mentioned previously will also result in a dramatically increased training time, and unfortunately time was not a luxury throughout the duration of this project.

### **7.4.3 Establishing a beginning and ending to each generated piece**

Currently once a sequence of notes is generated and ready to have its notes/chords extracted, the *random.randit()* function initializes note generation to begin at a random point within the sequence, therefore generating a unique sound each time the generation function is ran, but the drawbacks to this function is that there is no structural beginning or end to a musical piece, they typically begin and end abruptly which hinders the potential dramatic effect of a musical piece.

With the creation of new classes, it would be possible to identify the various aspects of each individual MIDI file such as the musical beginning to end that entails an introduction, melody, bridge, outro, etc. Thus, creating a more musically concise sound.

### **7.4.4 More in-depth objective evaluation methods**

The final improvements that could be made to the project as a whole is adding more depth to the evaluation process. Although subjective based means of evaluation are the most appropriate for evaluating a musical output, adding an objective based component to measure performance metrics to do with the musical structure of the generated pieces would add another level of depth and validity to the evaluation process as a whole.

Attempts were made at making use of an evaluation tool that was created by Yang et al. (2018), to measure these aforementioned performance based metrics such as rhythmic patterns, tonal distance, pitch classes, interval frequencies, etc.

Unfortunately, the documentation for the tool was poor and after several days of attempting to figure out how to implement it correctly there was no success and as a result alternate evaluation methods were created to assess the output of this project.

## 7.5 Evaluation of Personal Performance

This thesis has been the biggest piece of work I have ever produced, with a high level of technicality explored which has been one of the most difficult challenges I've experienced in my education, but it has been equally pleasant as it has been challenging, as I have been able to significantly increase my knowledge and expertise on the topic at hand whilst experiencing the joy of learning.

As a person with a musical background and passion for the creativity that exists within the music industry, I wanted to create a project that was a genuine reflection of the passions that exist within me. And throughout my experience of tertiary education, I have discovered a large interest and love for artificial intelligence and machine learning.

With the combination of these two passions the idea for this project was born during the time that I was brainstorming ideas for my bachelor's degree thesis, but I realized that during this time I did not have the knowledge and technical ability to implement such a task, and so the idea was put on halt. After the completion of my bachelor's degree and commencement of my master's degree, I obtained a wealth of knowledge on algorithms, data structures, and machine learning, and it was at this time that I finally realized the idea of a thesis enveloping two my major passions.

Once work on the project began, I quickly realized how difficult the topic was and that it would require an immense amount of time and focus to properly formulate an appropriate outcome. With the guidance of my supervisor Dr Vuong, he helped me to narrow my ideas and formulate them into something achievable within the deadline, and the process of implementation began.

A major setback that I faced was about 2 months before the final submission when I experienced system wide data loss whilst troubleshooting an issue with TensorFlow that was preventing me from training my model.

As a result, I tried to update my system version of Python on my Ubuntu operating system, but in doing so it affected all of Python's dependencies within Ubuntu that caused a corrupt operating system and system wide data loss. The work I had completed on my report was safe because I have dual booted both Windows 10 and Ubuntu 20.04 onto my computer. My Windows environment is where I type documents and this was unaffected, but I do all of my programming and development within Ubuntu, and I naively had not backed up my development.

The resulting affect was a complete restart on my implementation when I was already 1 month into development. I had made significant progress that required starting again, thankfully I knew how to retrace my steps as I had already achieved the work once, but it was still quite a large set back that caused extreme anxiety and overwhelming feelings of dread.

With the encouragement of my family and friends I was able to get back up and dust myself off to continue working hard on restoring the fragments of my project. I fortunately was able to bounce back and resume working diligently, this time backing up my work profusely.

As the work process continued, I had regular check-ins with my supervisor who advised me and guided me to achieving a successful outcome within the scope of my project. I managed my time well and put in a specific number of hours every week to ensure I was staying on top of project deadlines.

I am extremely satisfied with the final outcome of the project and all the work that has been achieved throughout the duration of this year. Although there are several means by which this project could be improved, as a novel implementation and proof of concept it performed what it set out to achieve and as a result, I am pleased to present this work as my greatest academical achievement to date.

## 8 References

- Alpern, A., 1995. Techniques for Algorithmic Composition of Music. [online] Citeseerx.ist.psu.edu. Available at: <<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9364&rep=rep1&type=pdf>> [Accessed 4 August 2021].
- Barker, A., 2010. Mathematical Beauty Made Audible: Musical Aesthetics in Ptolemy's Harmonics. [online] University of Chicago Press Journals logo. Available at: <<https://www.journals.uchicago.edu/doi/pdf/10.1086/657028>> [Accessed 16 August 2021].
- Encyclopedia Britannica. 2013. Chord | music. [online] Available at: <<https://www.britannica.com/art/chord-music>> [Accessed 13 August 2021].
- Encyclopedia Britannica. 2017. Octave | music. [online] Available at: <<https://www.britannica.com/art/octave-music>> [Accessed 13 August 2021].
- Encyclopedia Britannica. 2007. Key | music. [online] Available at: <<https://www.britannica.com/art/key-music>> [Accessed 13 August 2021].
- Encyclopedia Britannica. 2017. scale | Definition, Music Theory, & Types. [online] Available at: <<https://www.britannica.com/art/scale-music>> [Accessed 13 August 2021].
- Encyclopedia Britannica. 2019. pitch | Definition, Frequency, & Music. [online] Available at: <<https://www.britannica.com/art/pitch-music>> [Accessed 13 August 2021].

- St. George, B., 2019. What is the Turing Test? [online] SearchEnterpriseAI. Available at: <<https://searchenterpriseai.techtarget.com/definition/Turing-test>> [Accessed 17 August 2021].
- IBM.com. 2020. What are Neural Networks? [online] Available at: <<https://www.ibm.com/cloud/learn/neural-networks>> [Accessed 9 August 2021].
- Alyson McLamore., "Motif in Music ." New Dictionary of the History of Ideas. . Retrieved August 20, 2021 from Encyclopedia.com:  
<https://www.encyclopedia.com/history/dictionaries-thesauruses-pictures-and-press-releases/motif-music>
- Nattiez, J., 1993. Music and Discourse: Toward a Semiology of Music. [online] Semantic Scholar. Available at: <<https://www.semanticscholar.org/paper/Music-and-Discourse%3A-Toward-a-Semiology-of-Music-Nattiez/bbe9f2a4f31a2b0d41fab228205036c17417e339>> [Accessed 13 August 2021].
- Nicholson, C., 2020. A Beginner's Guide to LSTMs and Recurrent Neural Networks. [online] Pathmind. Available at: <<https://wiki.pathmind.com/lstm>> [Accessed 16 August 2021].
- Tch, A., 2017. The mostly complete chart of Neural Networks, explained. [online] Medium. Available at: <<https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>> [Accessed 11 August 2021].
- Yang, L. and Lerch, A., 2018. The Evaluation of Generative Models in Music. [online] Musicinformatics.gatech.edu. Available at: <[https://musicinformatics.gatech.edu/wp-content\\_nondefault/uploads/2018/11/postprint.pdf](https://musicinformatics.gatech.edu/wp-content_nondefault/uploads/2018/11/postprint.pdf)> [Accessed 5 August 2021].

## 9 Bibliography

- Medium. 2020. 10 Stages Of A Machine Learning Project In 2020 (And Where You Fit). [online] Available at: <<https://towardsdatascience.com/10-stages-of-a-machine-learning-project-in-2020-and-where-you-fit-cb73ad4726cb>> [Accessed 18 May 2021].
- Medium. 2020. How AI Is Shaping the Future of the Music Industry. [online] Available at: <<https://medium.com/swlh/how-ai-is-shaping-the-future-of-the-music-industry-f10d31f53be3>> [Accessed 19 May 2021].
- Chen, L., 2021. Generating Music Using LSTM Neural Network. [online] Medium. Available at: <<https://becominghuman.ai/generating-music-using-lstm-neural-network-545f3ac57552>> [Accessed 23 May 2021].
- Kotecha, N., 2018. Generating Music using an LSTM Network. [online] Arxiv.org. Available at: <<https://arxiv.org/ftp/arxiv/papers/1804/1804.07300.pdf>> [Accessed 23 May 2021].
- Pai, A., 2020. Automatic Music Generation | Music Generation Deep Learning. [online] Analytics Vidhya. Available at: <<https://www.analyticsvidhya.com/blog/2020/01/how-to-perform-automatic-music-generation/>> [Accessed 26 May 2021].
- Shah, F., 2019. LSTM Based Music Generation. [online] Ieeexplore.ieee.org. Available at: <<https://ieeexplore.ieee.org/document/8995760>> [Accessed 26 May 2021].



- Cope, D., 2000. The Algorithmic Composer. [online] Google Books. Available at: <[https://books.google.co.uk/books?hl=en&lr=&id=rFGH07I2KTcC&oi=fnd&pg=PR9&dq=the+algorithmic+composer&ots=FthXhjC13G&sig=tQHtLxSip2utmVDf4gO6\\_Cict0#v=onepage&q=the%20algorithmic%20composer&f=false](https://books.google.co.uk/books?hl=en&lr=&id=rFGH07I2KTcC&oi=fnd&pg=PR9&dq=the+algorithmic+composer&ots=FthXhjC13G&sig=tQHtLxSip2utmVDf4gO6_Cict0#v=onepage&q=the%20algorithmic%20composer&f=false)> [Accessed 30 May 2021].
- MBP. 2021. Project Risk Management - MBP. [online] Available at: <<https://www.mbpce.com/what-we-do/project-analysis-consulting/project-risk-management/>> [Accessed 31 May 2021].
- Pulver, A., 2016. LSTM with Working Memory. [online] Arxiv.org. Available at: <<https://arxiv.org/pdf/1605.01988.pdf>> [Accessed 11 August 2021].
- Skúli, S., 2017. How to Generate Music using a LSTM Neural Network in Keras. [online] Medium. Available at: <<https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5>> [Accessed 13 August 2021].
- Yang, L., 2017. MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION. [online] Arxiv.org. Available at: <[https://arxiv.org/pdf/1703.10847.pdf?source=post\\_page----->](https://arxiv.org/pdf/1703.10847.pdf?source=post_page----->) [Accessed 13 August 2021].
- Anderson, G., 1983. Pythagoras and the Origin of Music Theory on JSTOR. [online] Jstor.org. Available at: <[https://www.jstor.org/stable/24045969?seq=1#metadata\\_info\\_tab\\_contents](https://www.jstor.org/stable/24045969?seq=1#metadata_info_tab_contents)> [Accessed 16 August 2021].
- Masi, M., 1979. The Liberal Arts and Gerardus Ruffus' Commentary on the Boethian De Arithmetica. [online] Jstor.org. Available at: <[https://www.jstor.org/stable/2539405?seq=1#metadata\\_info\\_tab\\_contents](https://www.jstor.org/stable/2539405?seq=1#metadata_info_tab_contents)> [Accessed 16 August 2021].

- Yang, L. and Lerch, A., 2018. On the evaluation of generative models in music. [online] Musicinformatics.gatech.edu. Available at: <[https://musicinformatics.gatech.edu/wp-content\\_nondefault/uploads/2018/11/postprint.pdf](https://musicinformatics.gatech.edu/wp-content_nondefault/uploads/2018/11/postprint.pdf)> [Accessed 17 August 2021].
- Hewahi, N., 2018. Generation of music pieces using machine learning: long short-term memory neural networks approach. [online] Taylor & Francis. Available at: <<https://www.tandfonline.com/doi/full/10.1080/25765299.2019.1649972>> [Accessed 20 August 2021].
- Numpy.org. 2021. What is NumPy? — NumPy v1.21 Manual. [online] Available at: <<https://numpy.org/doc/stable/user/whatisnumpy.html>> [Accessed 22 August 2021].
- GeeksforGeeks. 2020. How to use Glob() function to find files recursively in Python? - GeeksforGeeks. [online] Available at: <<https://www.geeksforgeeks.org/how-to-use-glob-function-to-find-files-recursively-in-python/>> [Accessed 22 August 2021].
- Agrawal, A., 2014. Python pickling: What it is and how to use it securely | Synopsys. [online] Software Integrity Blog. Available at: <<https://www.synopsys.com/blogs/software-security/python-pickling/>> [Accessed 22 August 2021].
- Web.mit.edu. 2021. What is music21? — music21 Documentation. [online] Available at: <<https://web.mit.edu/music21/doc/about/what.html>> [Accessed 22 August 2021].
- TensorFlow. 2021. TensorFlow. [online] Available at: <<https://www.tensorflow.org/>> [Accessed 22 August 2021].

- Team, K., 2021. Keras: the Python deep learning API. [online] Keras.io. Available at: <<https://keras.io/>> [Accessed 22 August 2021].
- M, S., 2019. WHAT IS A PIPELINE IN MACHINE LEARNING? HOW TO CREATE ONE?. [online] Medium. Available at: <<https://medium.com/analytics-vidhya/what-is-a-pipeline-in-machine-learning-how-to-create-one-bda91d0ceaca>> [Accessed 24 August 2021].
- Medium. 2020. A Complete Guide to Adam and RMSprop Optimizer. [online] Available at: <<https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be>> [Accessed 27 August 2021].
- 8notes.com. 2019. Improvising on the Piano with the Pentatonic Scale - 8notes.com. [online] Available at: <<https://www.8notes.com/school/lessons/piano/improvising-on-the-piano-with-the-pentatonic-scale.asp>> [Accessed 30 August 2021].
- Eck, D. and Schmidhuber, J., 2002. A First Look at Music Composition using LSTM Recurrent Neural Networks. [online] People.idsia.ch. Available at: <<https://people.idsia.ch/~juergen/blues/IDSIA-07-02.pdf>> [Accessed 5 September 2021].

## 10 Appendices

### 10.1 Appendix A – Musical Excerpt from the Listening-Based Experiment

[https://www.youtube.com/watch?v=BH8OF1bdfJA&ab\\_channel=JJRoses-Agoro](https://www.youtube.com/watch?v=BH8OF1bdfJA&ab_channel=JJRoses-Agoro)

### 10.2 Appendix B – Alternate MIDI output that was generated by the Network

[https://www.youtube.com/watch?v=bekICqZ7cHY&t=2s&ab\\_channel=JJRoses-Agoro](https://www.youtube.com/watch?v=bekICqZ7cHY&t=2s&ab_channel=JJRoses-Agoro)

### 10.3 Appendix C – Sample from the Dataset

[https://www.youtube.com/watch?v=QE1CFH5HwYs&ab\\_channel=JJRoses-Agoro](https://www.youtube.com/watch?v=QE1CFH5HwYs&ab_channel=JJRoses-Agoro)

### 10.4 Appendix D – File structure of the Implementation









| Name  | Size     | Modified | Star |
|---|----------|----------|------|
|  data                | 1 item   | 12 Aug   | ☆    |
|  midi_output         | 20 items | 7 Sep    | ☆    |
|  midi_songs          | 80 items | 7 Sep    | ☆    |
|  Saved Model Weights | 1 item   | 14:42    | ☆    |
|  Sheet Music         | 2 items  | 14:29    | ☆    |
|  Tests               | 4 items  | 14:29    | ☆    |
|  Prediction.ipynb    | 10.2 kB  | 7 Sep    | ☆    |
|  Training.ipynb      | 73.8 kB  | Mon      | ☆    |

Figure 35 - File structure of the Implementation

- **data:**  
Note/Chords in binary format
- **midi\_output:**  
MIDI files generated by the network
- **midi\_songs:**  
Dataset consisting of MIDI files
- **Saved Model Weights:**  
The saved weight from the final epoch of training
- **Sheet Music:**  
The sheet music of the musical piece that was evaluated and a sample from the dataset
- **Tests:**  
Python notebooks where development debugging was conducted
- **Prediction:**  
The notebook where predictions were made based on the dataset
- **Training:**  
The notebook where training was conducted based on the dataset