Department of Computer Science & Engineering

A Project Report on

Music Generation using Deep Learning Techniques

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering in Computer Science & Engineering

By

Prashant Krishnan V	1MS15CS091
R Deepak	1MS15CS095
Venkat Krishnamohan	1MS15CS139
Vivek Chandra Sheel	1MS15CS143

Under the guidance of

Dr. S. Rajarajeswari
Associate Professor

M S RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)
BANGALORE-560054
www.msrit.edu
May 2019

RAMAIAH Institute of Technology

Department of Computer

Science & Engineering

CERTIFICATE

Certified that the project work entitled "**Music Generation using Deep Learning Techniques**" carried out by Raji Subramanian – 1MS15CS000 a bonafide student of M.S.Ramaiah Institute of Technology Bengaluru in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgavi during the year 2018-19. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Project Guide **Head of the Department**

DR. S. RAJARAJESWARI **DR. ANITA KANAVALLI**

External Examiners

Name of the Examiners: **Signature with Date**

1.

2.

RAMAIAH
Institute of Technology

Department of Computer Science & Engineering

DECLARATION

We, hereby, declare that the entire work embodied in this project report has been carried out by us at M.S.Ramaiah Institute of Technology, Bengaluru, under the supervision of **Dr. S. Rajarajeswari, Associate Professor,** Dept of CSE. This report has not been submitted in part or full for the award of any diploma or degree of this or to any other university.

Signature	Signature
RAJI SUBRAMANIAN	RAJI SUBRAMANIAN
1MS15CS000	1MS15CS000

Signature	Signature
RAJI SUBRAMANIAN	RAJI SUBRAMANIAN
1MS15CS000	1MS15CS000

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We would like to express our profound gratitude to the Management and **Dr. N.V.R Naidu** Principal, M.S.R.I.T, Bengaluru for providing us with the opportunity to explore our potential.

We extend our heartfelt gratitude to our beloved **Dr. Anita Kanavalli**, HOD, Computer Science and Engineering, for constant support and guidance.

We wholeheartedly thank our project guide **Dr. S. Rajarajeswari ,** for providing us with the confidence and strength to overcome every obstacle at each step of the project and inspiring us to the best of our potential. We also thank her for her constant guidance, direction and insight during the project.

This work would not have been possible without the guidance and help of several individuals who in one way or another contributed their valuable assistance in preparation and completion of this study.

Finally we would like to express sincere gratitude to all the teaching and non-teaching faculty of CSE Department, our beloved parents, seniors and my dear friends for their constant support during the course of work.

RAJI SUBRAMANIAN RAJI SUBRAMANIAN

RAJI SUBRAMANIAN RAJI SUBRAMANIAN

Abstract

Music is essentially a sequence of sounds, with a frequency, volume and timbre, organised over time. The right combination of musical notes and rhythm can evoke emotions in the listener. This property makes music, like all other art forms, inherently human. Recent advances in deep learning, however, have broadened the spectrum of human aspects that can be automated such as the ability to translate languages, identify objects visually, and analyse audio, which begs the question of whether Artificial Intelligence (AI) can create music. This project aims to develop models that employ deep learning techniques to generate human like music. The model explores two deep learning techniques, Long Short Term Memory (LSTM) and Generative Adversarial Networks (GANs).

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 General Introduction

Artificial Intelligence has been pushing boundaries in various fields, generating human-like music is one among them. Deep learning has recently been at the forefront of many innovations in the technology industry like image recognition, voice detection, language processing, etc. Music is something that transcends all barriers in society. There have been many attempts of combining various forms of art with AI and music generation is one among them. The value of music in an individual's life is multifold. It is one of the arts that emotionally connect with humans, whether it is being happy, sad, calm, angry, etc. It also has a therapeutic value to it.

This paper aims to generate musical sequences using deep learning techniques. The key goal is for the model to learn musical styles, and be able to generate new musical content. This is not a trivial task even for a machine as it has to understand the underlying sub structure of a particular musical piece and then create a sensible, cohesive sequence. The task of learning musical patterns is an unsupervised one. The model tries to look for patterns in music that makes sense, that it understands and subsequently generates. Before the advent of machine learning techniques, there have been various approaches in generating music which involves only a probabilistic approach using certain rules. These rules could be related to music theory or specific to how the person would want the machine to create the music. With the ability to develop complex artificial neural networks, the task of understanding the sequences of chords and notes in music has become easier.

We limit our study with generation of music using a single instrument. We primarily explore the concepts of Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTMs) and Generative Adversarial Networks (GANs) to generate our music. The input music files are of Musical Instrument Digital Interface (MIDI) format which is a standard in the music industry. A library Music21 is used to be able to parse these MIDI files and use them in a machine readable format. We also aim to get feedback of the music from different types of people, i.e. ones who have good knowledge of music, average knowledge of music and not a lot of knowledge of music. The survey would aim on understanding if people can distinguish human created and machine generated music. This would help us evaluate the model with respect to how close to human like music its generated music is.

The model developed in this paper could help a musician in creating his/her music. In the future, a lot of the music made by musicians could be assisted by AI. The paper is divided into X sections talking about related work in this field, the methodology used, model, conclusion and future work.

1.2 Problem Statement

Deep learning in the past decade has had extensive usage in various fields like financial fraud detection, recommendation systems, automatic speech recognition, etc. But, it is still at its nascent stage in other fields which are more inherent to humans' creative abilities like art, music, poetry, etc.

This project aims to process a dataset consisting of music files, learn the underlying patterns, and develop a generative AI model which can produce human like music.

1.3 Objectives of the Project

The primary objective of the project is to create a generative AI model, using Deep Learning techniques, that creates good music. The model will initially generate plain melodies on a single instrument, and will then be built on to integrate chords, harmonies, and finally multiple instruments. This project is a step in the direction of understanding how the human brain interprets the scientific theory of music by analysing how an AI model does the same.

1.4 Project Deliverables

- To be able to parse MIDI files to its components
- To learn underlying patterns in the music sequences
- Developing models which can generate novel sequences based on deep learning techniques like LSTMs and GANs
- Comparing the samples generated by the different neural network models
- To generate music samples of a specified duration and beats per minute (BPM)

1.5 Current Scope

The purpose of this project like mentioned above is to be utilize deep learning techniques to generate music. The system aims at using various deep learning concepts like RNNs, LSTMs, GANs, etc. to be able to create melodies. AI has had major impacts in various fields over the years. Medical diagnosis, stock trading, e-commerce and several other fields have had large advancements and big challenges solved thanks to AI. Like other advanced AI use cases, this technology could have a few in the coming years. A musician facing something on the lines of a writer's block or increased personalised experiences of music based on mood are a few future applications.

1.6 Future Scope

This is one of the applications that is pushing the boundaries of AI especially in the creative fields. The future scope of the project includes training and generating multi instrument music. This would mean, models learning individual components of the music and then correspondingly generating music.

With greater computational power, the input data can be vast and can comprise of different genres as well. That way, the training for larger datasets could This would mean, it could help various musicians around the world. The help could be giving them ideas to further their creative juices.

Chapter 2

Project Organisation

2.1 Software Process Models

The chosen software process model is Agile Methodology. This methodology is advantageous to us as it allows for flexibility for change and for improvement.

It has a lower cost of working. This method encourages open communication among team members, and clients at every step of software development. It provides a competitive advantage by catching defects and making changes throughout the development process, instead of at the end. Speeds up time spent on evaluations since each evaluation is only on a small part of the whole project. Ensures changes can be made quicker and throughout the development process by having consistent evaluations to assess the product with the expected outcomes requested. It keeps each project transparent by having regular consistent meetings with the clients and systems that allow everyone involved to access the project data and progress.

2.2 Roles and Responsibilities

Literature Survey	Vivek, Prashant, Deepak, Venkat
Domain Analysis	Vivek, Prashant, Deepak, Venkat
Synopsis	Venkat, Prashant
System Requirements	Vivek, Prashant
Project Planning	Deepak, Venkat
Design of system architecture	Venkat, Deepak, Prashant
Coding	Vivek, Prashant, Deepak, Venkat

Testing	Vivek, Prashant, Deepak, Venkat
Report	Vivek, Prashant, Deepak, Venkat

Chapter 3

Literature Survey

3.1 Introduction

3.2 Related Works with the citation of the References

The paper SampleRNN: An Unconditional end-to-end Neural Audio Generation Model et al. [1], proposes a novel model for unconditional audio generation based on generating one audio sample at a time. The model, which profits from combining memory-less modules, namely autoregressive multilayer perceptrons, and stateful recurrent neural networks in a hierarchical structure is able to capture underlying sources of variations in the temporal sequences over very long time spans, on three datasets of different nature

In the groundbreaking paper Generative Adversarial Nets [2], Ian J. Goodfellow et al. proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency.

In the paper, Generating Sequences With Recurrent Neural Networks [3] by Alex Graves from University of Toronto, we see how Long Short-Term Memory (LSTM) recurrent neural network works can be used to generate complex sequences with long-range structure, simply by predicting one data point at a time. The system is further extended to handwriting synthesis. It does this by conditioning its predictions on a text sequence. The paper demonstrates the ability of LSTM recurrent neural networks to generate both discrete and real valued sequences with complex, long range structure. We use the concepts that the author demonstrates for text and online handwriting to generate sequences of notes for music.

Douglas Eck et al., in their study Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks [4], show that LSTM is also a good mechanism for learning to compose music. They presented experimental results showing that LSTM successfully learns a form of blues music and is able to compose novel melodies in that style. They observed that once the network found the relevant structure it does not drift from it: LSTM was able to play the blues with good timing and proper structure.

In Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription [5], Nicolas Boulanger-Lewandowskiwe et al., demonstrated that the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) outperforms many traditional models of polyphonic music. They have introduced a generalization of the RTRBM, called the RNNRBM, that allows more freedom to describe the temporal dependencies involved. The model learns harmonic and rhythmic probabilistic rules from polyphonic music scores of varying complexity, substantially better than popular methods in music information retrieval.

Darrel Conklin from City University in London wrote a paper titled, Music Generation from Statistical Models [6]. In this paper, the problem of generating music is equated to a statistical problem of sampling from a statistical model. It uses various statistical concepts like random walk, HMMs (Hidden Markov Model), stochastic sampling and pattern-based sampling. The objective that the author took was to construct models for a stylistic corpus that assigns high probability to new pieces in the style. The drawback in this approach is that only taking statistical approaches misses out on key features of the music samples which can be achieved through deep learning. They have rephrased the problem as a classic search and sampling problem which is why it has many drawbacks.

We look at a paper written by Yoon Kim et al.. [7] which focuses on Character-Aware Neural Language Models. This is a slightly different domain that the authors are dealing with, as their task is more in line with Natural Language Processing (NLP) with applications in speech recognition and text generation. We only focus on the text generation part. They use Long Short-Term Memory baselines for generation of text. These concepts are similar to some of the tasks in our paper as we look to generate music from a sheet music input or a textual input. Their language model uses RNNs and also character level CNN to analyze the language model.

In a paper written by Soroush Mehri et al.. [8], they develop an end-to-end model which generates audio unconditionally. They use and combine memory-less modules, namely autoregressive multilayer perceptrons, and stateful recurrent neural networks in a hierarchical structure to capture variations that exist in temporal sequences. They also prefer human evaluation of models over competing models. They try to focus only on one aspect of audio input which is able to produce just one audio sample. The model also requires to learn representation of sequential data with high temporal resolution and long-range complex structure.

Another paper Deep Learning for Music written by Allen Huang et al.. [9] has tried to create a deep learning model that generates music that has both the qualities of harmony and melody. They have tried to perform end-to-end learning and generation with deep neural nets alone. It shows that a multi-layer LSTM, character-level language model when applied to two separate data representations is capable of generating music. The model described in the paper doesn't capture the underlying melodic structure.

In the paper A First Look at Music Composition using LSTM Recurrent Neural Networks by DouglasEck et al. [10] proposed using a Long-Short-Term Memory (LSTM) model for learning to compose music. The model tries to overcome the failure of other Recurrent Neural Network models that cannot keep track of temporally distant events that indicate global music structure. The LSTM model successfully learnt the chord structure and was able to generate new instance of musical form. This enables the creators to produce music that never diverges a long way from the first harmony movement melody. In any case, this architecture trains on a set number of harmonies and can't make an increasingly diverse combination of notes.

In the paper Text-based LSTM networks for Automatic Music Composition by Keunwoo Choi, George Fazekas, and Mark Sandler et al. [11] The proposed network is designed to learn relationships within text documents that represent chord progressions and drum tracks in two case studies. In the experiments, word-RNNs (Recurrent Neural Networks) show good results for both cases, while character-based RNNs (char-RNNs) only succeed to learn chord progressions. The proposed system can be used for fully automatic composition or as semi automatic systems that help humans to compose music by controlling a diversity parameter of the model.

Then again Boulanger-Lewandowski et al.. [12] endeavor to manage the test of learning complex polyphonic structure in music. They utilize a repetitive worldly confined Boltzmann machine(RTRBM) so as to display unconstrained polyphonic music. Utilizing the RTRBM engineering enables them to speak to a convoluted conveyance over each time step as opposed to a solitary token as deepest character language models. This enables them to handle the issue of polyphony in produced music.

In this paper [13], Debora et al. compare the learning ability of two different algorithms BPTT (Back-Propagation Through Time) and LSTM (Long - Short Term Memory). They use 2 different representation of data for training the models namely interval representation and circle-of-thirds representation. The final composition strategy was based on statistical information about the training set, which gives the information about the probability of the actual pitch based on the occurrence of the previous one. After testing the two methods it was determined that LSTM could learn the patterns better even though they were separated over time.

In this paper [14], Huanru et al., propose a method by which they can tune the output generated by a model. Piano roll is the chosen data format for training the data. They proposed a method where they used biaxial LSTM since the model must condition the probability of playing a note along two axes: time and note. The proposed model uses style conditioning at every layer to learn information about each style of music.

Gino et al., propose a novel approach to generate polyphonic music using multiple lstms to create pleasing and harmonic sounds[15]. The method uses multiple LSTMs to achieve the desired result. Model uses chord LSTM which learns a meaningful representation of chords from the training data. Polyphonic LSTM, takes the piano rolls and generated chord progression from the chord LSTM as the inputs. The final LSTM produces the final music as a piano roll and the result obtained is largely harmonic.

In this paper [16], Maria et al., explore the possibility of music generation using LSTMs and MIDI music format. The main objective achieved is learning how to generate the two fundamental musical expression elements: dynamics and expressive tempo. The proposed method uses two LSTMs: dynamics model that can generate velocity values (loudness of note) and tempo model that can generate absolute global tempo values. The model was capable of recognizing a few dependencies and match real performances to some extent accurately.

3.3 Conclusion of Survey

SL No.	Title & Author	Techniques Used	Results	Discussion / Future work/ Advantage/ Disadvantage
1.	SampleRNN: An Unconditional	Recurrent Neural Network (RNN)	Works in a hierarchica	It combines memory-less

	end-to-end Neural Audio Generation Model		l structure, and captures underlying sequences in temporal sequences	modules. However, long temporal structures don't perform well
2.	Ian Goodfellow - Generative adversarial nets. In Advances in Neural Information Processing Systems	Generative Adversarial Networks (GANs)	New framework using two networks, a generator and adversarial .	Produces generative samples in an unsupervised method.
3.	Generating Sequences With Recurrent Neural Networks, Alex Graves	Long Short-Term Memory (LSTMs)	Generates sequences like cursive handwritin g in various styles	Works for discrete sequences. Insight for internal representations can be better.
4.	Finding Temporal Structure in Music: Blues Improvisation with LSTM	Long Short-Term Memory (LSTMs) and Recurrent Neural Networks (RNNs)	LSTM could capture the long term	Comparison of performances is something that lacks.

	Recurrent Networks, Douglas Eck and JÄurgen Schmidhuber		structure of the musical style - Blues.	Also, a more methodical approach to the parameters could be taken.
5.	Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription - Nicolas Boulanger-Lewandowski, Yoshua Bengio, Pascal Vincent	RNN-based Restricted Boltzmann Machine	Improved prediction accuracy of models by employing strategies related to the description of temporal dependencies.	The model serves as a symbolic prior for polyphonic transcription. However, longer term musical structure remains elusive in the approach.
6.	Music Generation from Statistical Models - Darrell Conklin	Explores several statistical sampling methods like Random walk method, Hidden Markov models and Viterbi decoding, Stochastic sampling, and Pattern-based sampling	The models were flawed for generating complete pieces. Pattern continuatio	The author suggests modern pattern discovery methods be applied to extant pieces to reveal their repetition

			n could be handled, but there was no way to specify at the outset of generation where repeated patterns should begin and end.	structure and provide musical cohesion to new productions.
7.	Character-aware neural language models - Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush	Deep learning model which uses CNN, and a highway network over characters whose output is fed to an LSTM based Language Model.	Achieved performance levels on par with state-of-the-art approaches, having approximately 60%fewer parameters.	The model could be able to encode, from characters only, rich semantic and orthographic features.

8.	Learning to forget: Continual Prediction with LSTM	Upgraded RNN to LSTM	Solved the problem of vanishing gradients and made model capable of rememberi ng sequences over time.	LSTM has become the preferred model for learning sequential data
9.	Deep Learning for Music - Allen Huang, Raymond Wu	2-layered Long Short Term Memory (LSTM) recurrent neural network (RNN) architecture	Models were able to learn meaningfu l musical structure and an average rating of 7.0±1.87, 5.3±1.7, and 6.2±2.4	The model described in the paper doesn’t capture the underlying melodic structure.
10.	A First Look at Music Composition using LSTM Recurrent Neural	LSTM	Successful ly learned the global structure	Preliminary Experiments that can be further

	Networks - Douglas Eck, Jurgen Schmidhuber		of a musical form, and used that informatio n to compose new pieces in the form.	advanced to get better results.
11.	Text-based LSTM networks for Automatic Music Composition - Keunwoo Choi, George Fazekas, Mark Sandler	word based LSTM, Char RNN	It learns relationshi ps within text documents that represent chord progressio ns and drum tracks	Char-RNNs turned out to fail to learn the drum tracks and output arbitrary 0's and 1's without any structures.
12.	Modeling Temporal Dependencies in High-Dimensional Sequences:Applicat ion to Polyphonic Music Generation	Confined Boltzmann machine(RTRBM)	Observed an improvem ent in absolute accuracy	It can learn harmonic and rhythmic probabilistic rules from polyphonic

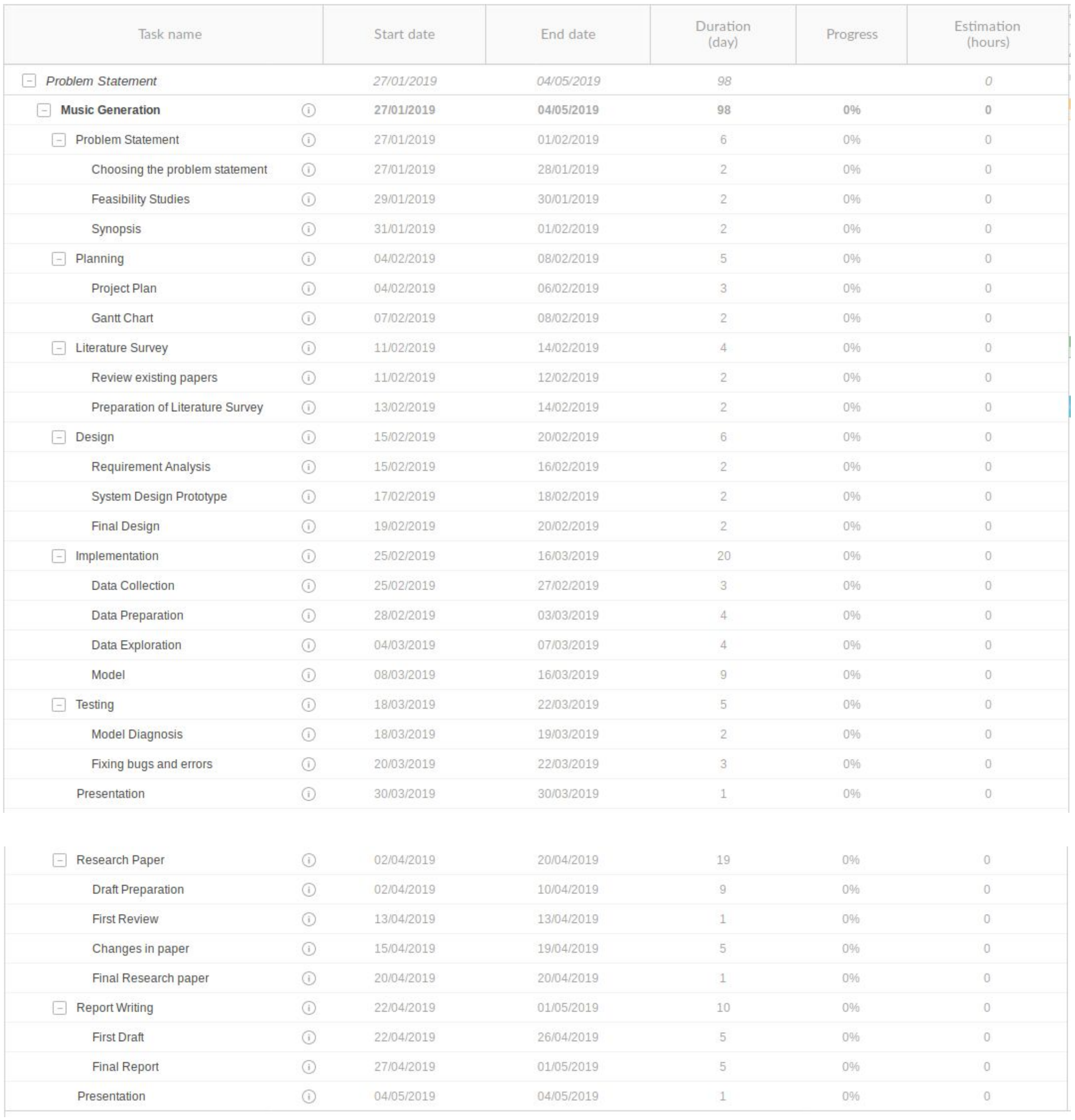
	and Transcription - Nicolas Boulanger-Lewandowski, Yoshua Bengio, Pascal Vincent		between 1.3% and 10% over the HMM approach.	music scores of varying complexity, substantially better than popular methods in music information retrieval.
13.	Composing Music with BPTT and LSTM Networks: Comparing Learning and Generalization Aspects - Débora C Corrêa, José HSaito, Sandra Abib	Backpropagation Through Time (BPTT), Long Short Term Memory (LSTM)	LSTM performed better than BPTT. It more readily learnt patterns which were separated over time.	LSTM is the prefered Neural Network structure to be used if patterns are to learnt over time.
14.	DeepJ: Style-Specific Music Generation - Huanru Henry Mao, Taylor Shin, Garrison W Cittrell	Biaxial LSTM	Model solved style consistency problem, making the	The ability to tweak certain parameters and obtain a desired style of music is highly

			output more accurate.	desirable.
15.	JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs - Gino Brunner, Yuyi Wang, Roger Wattenhofer and Jonas Wiesendanger	Multiple LSTMs	Output retains many characteristics from training data. The audio produced is harmonious and pleasing to hear.	Extension for including style or genre based learning.
16.	Generating musical expression of MIDI music with LSTM neural network - Maria, Adrian, Bartlomiej	2 LSTMs for dynamics and expressive tempo respectively	Created music which is on par which humans but only on a few training samples	Creating music which can resemble a real person is extremely hard as it has to learn all the intricate patterns which make music so wonderful.

Chapter 4

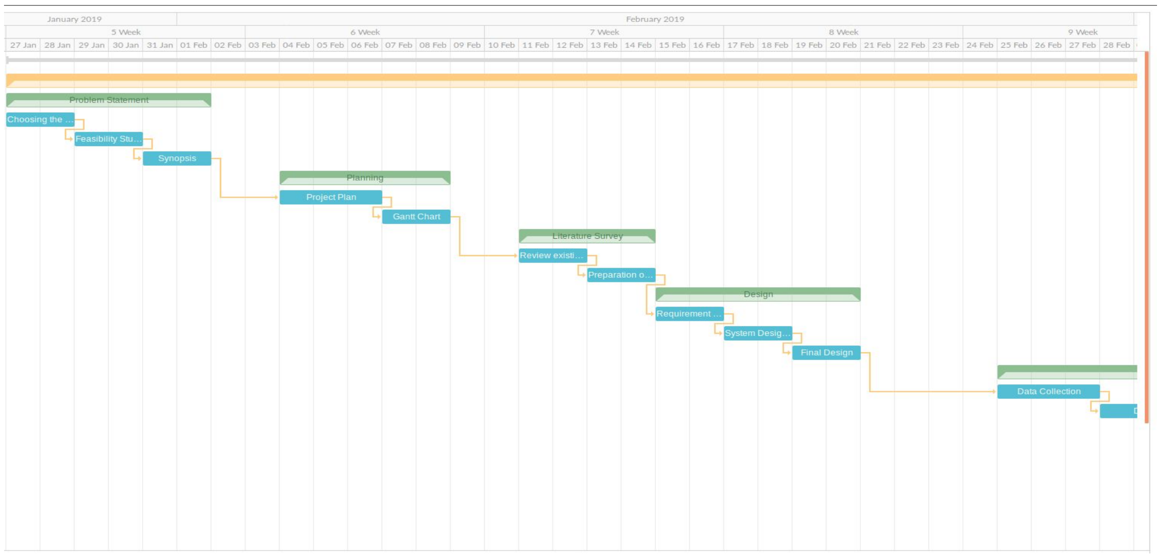
Project Organisation

4.1 Schedule of the Project

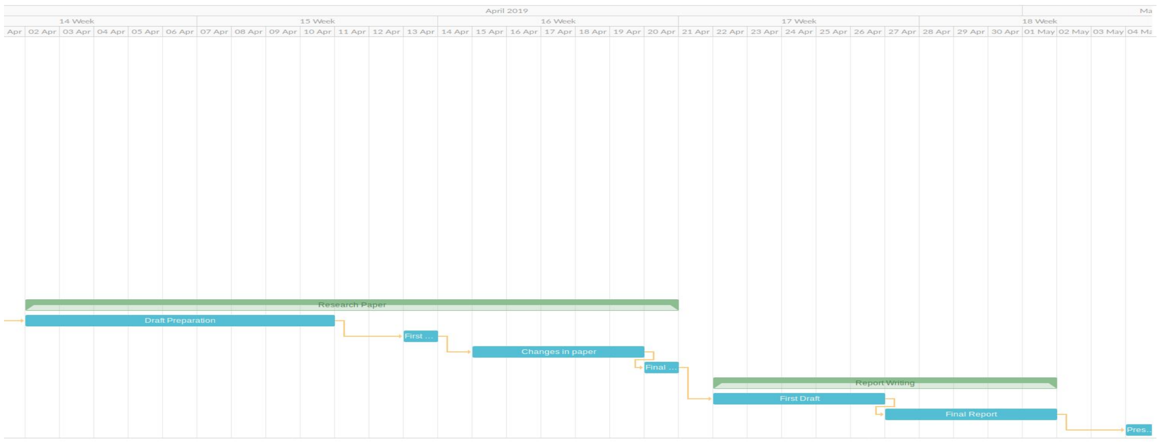


Task name	Start date	End date	Duration (day)	Progress	Estimation (hours)
Problem Statement	*27/01/2019*	*04/05/2019*	*98*		*0*
Music Generation	**27/01/2019**	**04/05/2019**	**98**	**0%**	**0**
Problem Statement	27/01/2019	01/02/2019	6	0%	0
Choosing the problem statement	27/01/2019	28/01/2019	2	0%	0
Feasibility Studies	29/01/2019	30/01/2019	2	0%	0
Synopsis	31/01/2019	01/02/2019	2	0%	0
Planning	04/02/2019	08/02/2019	5	0%	0
Project Plan	04/02/2019	06/02/2019	3	0%	0
Gantt Chart	07/02/2019	08/02/2019	2	0%	0
Literature Survey	11/02/2019	14/02/2019	4	0%	0
Review existing papers	11/02/2019	12/02/2019	2	0%	0
Preparation of Literature Survey	13/02/2019	14/02/2019	2	0%	0
Design	15/02/2019	20/02/2019	6	0%	0
Requirement Analysis	15/02/2019	16/02/2019	2	0%	0
System Design Prototype	17/02/2019	18/02/2019	2	0%	0
Final Design	19/02/2019	20/02/2019	2	0%	0
Implementation	25/02/2019	16/03/2019	20	0%	0
Data Collection	25/02/2019	27/02/2019	3	0%	0
Data Preparation	28/02/2019	03/03/2019	4	0%	0
Data Exploration	04/03/2019	07/03/2019	4	0%	0
Model	08/03/2019	16/03/2019	9	0%	0
Testing	18/03/2019	22/03/2019	5	0%	0
Model Diagnosis	18/03/2019	19/03/2019	2	0%	0
Fixing bugs and errors	20/03/2019	22/03/2019	3	0%	0
Presentation	30/03/2019	30/03/2019	1	0%	0
Research Paper	02/04/2019	20/04/2019	19	0%	0
Draft Preparation	02/04/2019	10/04/2019	9	0%	0
First Review	13/04/2019	13/04/2019	1	0%	0
Changes in paper	15/04/2019	19/04/2019	5	0%	0
Final Research paper	20/04/2019	20/04/2019	1	0%	0
Report Writing	22/04/2019	01/05/2019	10	0%	0
First Draft	22/04/2019	26/04/2019	5	0%	0
Final Report	27/04/2019	01/05/2019	5	0%	0
Presentation	04/05/2019	04/05/2019	1	0%	0

Fig. 4.1 Schedule of the project



(a)



(b)

Fig. 4.2 Gantt chart

4.2 Risk Identification

	Risk	**Probability**	**Impact**	**Exposure**	**Mitigation Plan**
1	Improper Dataset Absence of piano components in the samples of the input data	Medium	Medium	Low	Proper methods of data collection or use a reliable source for obtaining the dataset.
2	Non convergence of model During training, the loss function fails to converge to a minima value which results in poor output samples	Low	High	High	Modification of model architecture to better reflect the complexity of the input. Tune model hyperparameters to better suit the network

3	Overfitting to training set The output samples generated correspond too closely or exactly with a particular set of input data due to a modelling error.	Medium	Medium	Medium	Use of regularisation techniques like dropout to improve generalisation of model.
4	Insufficient data Deep Learning models require larger datasets in order to make accurate inference.	Low	Low	Medium	Algorithms that require less training data can be implemented. Robust training algorithms.
5	Lack of computational power Deep learning models require massive amounts of computational power to train neural networks and learn representations.	Medium	Medium	Low	Use faster implementations in the algorithms which use GPU-accelerated libraries. Choose values of batch size and schedule the jobs in the processor effectively

Chapter 5

Software Requirements Specification

5.1 Product Overview

Purpose

The purpose of this project is to be able to use various deep learning techniques to generate music. The field of audio processing and generation of sequences of text, audio and images have had great developments with the advent of deep learning. It is using these techniques that we aim to be able to generate musical melodies and sequences. We aim to see how various techniques fare at performing this task.

The purpose of this project like mentioned above is to be utilize deep learning techniques to generate music. The system aims at using various deep learning concepts like RNNs, LSTMs, GANs, etc. to be able to create melodies. AI has had major impacts in various fields over the years. Medical diagnosis, stock trading, e-commerce and several other fields have had large advancements and big challenges solved thanks to AI. Like other advanced AI use cases, this technology could have a few in the coming years. A musician facing something on the lines of a writer's block or increased personalised experiences of music based on mood are a few future applications.

Product Perspective

The product, as mentioned above, is an AI model that utilizes deep learning techniques to produce quality music. The model trains on multiple music samples, learns any new and essential features extracted from the samples, apart from the music theory knowledge base integrated into the model, and generates musical sequences of a fixed length, key, and rhythm.

Product Features

The primary feature of the product is to generate quality music of a given length, key, and rhythm.

User Classes and Characteristics

The product, being a generative AI model, is essentially passive and not interactive. Hence, there is only one class of users which is inclusive of everybody, irrespective of their musical knowledge or expertise. The product does, however, aid musical artists and music theorists.

Operating Environment

The product is designed to operate on UNIX based systems, such as Ubuntu.

Design and Implementation Constraints

The Design and Implementation Constraints are as follows:

1) Availability of a high end CUDA capable GPU for training purposes.
2) Availability of a high end processor and RAM.
3) Workability of a high quality MIDI player and ABC/Piano Roll to MIDI converter.

Assumptions and Dependencies

1. It is assumed that there is a large corpus of ABC notation music samples.
2. Dependencies are a variety of Python libraries which include the following:
    - Numpy
    - Pandas
    - Tensorflow
    - Keras

5.2. External Interface Requirements

5.2.1. User Interface

Front-end : A web page

- Inputs : The front end for user interface will have a front-end of a webpage through which user can either provide several inputs required for the model in the text input fields or an MIDI file.
- Output : The output will be a wav audio file generated by the model.

Back-end: A backend server which can interact with the neural network model to provide the corresponding output to the front-end web page.

5.2.2. Hardware Interface

Hardwares required for this music generation project are as follows:

- 32-bit or 64-bit Computer
- Operating System: Windows / Linux / MacOS
- Minimum 8GB of free disk space
- RAM: Minimum of 8GB
- GPU : Titan XP/ GTX - 1080/980/
- CPU : Intel® Core™ i5 processor onwards
- CUDnn

5.3.3. Software Interface

Softwares and libraries used in this project are mentioned below:

Software Used	**Description**
1. Anaconda	Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific

	computing, that aims to simplify package management and deployment.
2. CUDA	CUDA is a parallel computing platform and application programming interface model created by Nvidia. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit for general purpose processing
3. Jupyter Notebook	The Jupyter Notebook is an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
4. Keras	Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
5. Tensorflow	TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.
6. Python 3.x	Python is an interpreted, high-level, general-purpose programming language

7. Pandas, numpy	Pandas is a software library written for the Python programming language for data manipulation and analysis. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices,
8. Music21	Music21 is a Python-based toolkit for computer-aided musicology. People use music21 to answer questions from musicology using computers, to study large datasets of music, to generate musical examples, to teach fundamentals of music theory, to edit musical notation, study music and the brain, and to compose music (both algorithmically and directly).
9. Angular	Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.
10. Node	**Node.js** is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content *before* the page is sent to the user's web browser

5.2.4 Communications Interface

The project will use any web browser for all the communication between the frontend and backend part of the modal.

5.3 Functional Requirements

5.3.1 Audio to Text Conversion

The input to the model can either be textual ABC notation of music or an audio MIDI form. If an audio form is provided as input then an audio to text convertor is used to convert the MIDI form of music into textual ABC notation which can then be feeded into the network.

5.3.2 Text to Audio Conversion

The ABC notation produced as output by the model will then be converted into audio form using an text to audio convertor for music so that the music produced by model can be listened to and then used for providing feedback to the model.

5.3.3 Parsing MIDI files using Music21

A library Music21 is used to be able to parse these MIDI files and use them in a machine readable format. The file is broken down into its constituent components of chords and notes.

5.3.4 LSTM Functionality

LSTM will be utilized to remember patterns in time, and generate the next element in the sequence accordingly.

5.3.5 GAN Functionality

GAN shall be utilized comparatively against LSTM. GANs consist of two networks - the generator and the adversary, one pitting against the other. The generator, generates new data instances, while the other, the discriminator/adversary, evaluates them for authenticity

5.3.6 Penalty Function

Penalty function is essentially the loss function utilized for training the generative model parameters, which will enhance the quality of music produced.

5.3.7 Generation of MIDI file

The model should learn to distinguish between producing a good quality music instead of random noises. The feedback provided by the agent can be used to train the model to filter out noise and produce quality music.

5.3.8 MIDI to WAV conversion

The output midi file generated by the model is converted to wav format so that it is playable by the browser.

5.4 Nonfunctional Requirements

5.4.1 Performance Requirements

The steps involved to perform the implementation of music generation model are:

A) **Formatting the music to trainable data**

   The data obtained must be converted into a format which can be understood by the model. In our case we must convert the music to a piano roll format, which is a format indicating which notes are on and off over time.

B) **Fast training of the model**

   The model must be able to learn the patterns from the data as quickly and accurately as possible.

5.4.2 Software Quality Requirements

- **Response time:** The model should be able to able to run the training of the model and hence produce the output within a short period of time.
- **Correctness:** The model should create an output which is mostly similar to whatever input music it was fed.
- **Stability:** The model should not produce vastly different outputs over the same input.

Chapter 6

Design

6.1 Introduction

This System Design Document has been made to layout the proposed framework structure for Music generation by deep learning. This document provides an overview of how the Neural networks are used to generate good quality music from a small segment of music. A system architecture is included to show the high level design of the whole model and its working and several low level designs to show the detailed working of the individual components of the model.

6.2 Architecture Design

6.2.1 High Level System Design

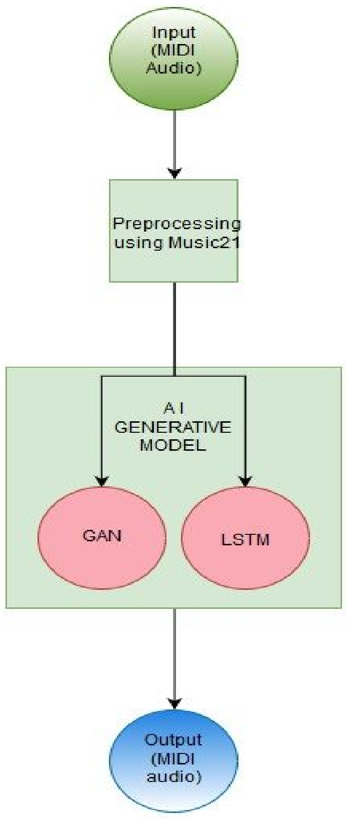


Fig. 6.1 System Architecture

The model architecture, as shown in Fig. 1, consists of 3 stages - Input, Model and Output.

- The input stage involves parsing of the input music files and extracting the musical elements, namely notes and chords. Model input sequences are also prepared in this stage.
- The Model stage has 2 phases - Model Training, and Model prediction
    - Model training involves creation of the deep learning model, initialising its weights during training, fitting it to the dataset, and storing weights.
    - Model prediction involves loading the trained weights, and predicting novel music based on input parameters - input sequence, sequence length and bpm.
- In the output stage, the generated sequences are added to a Midi stream, and is exported as a Midi file, which is then accessed by a webpage.

6.2.2 Low Level System Design (LSTM)

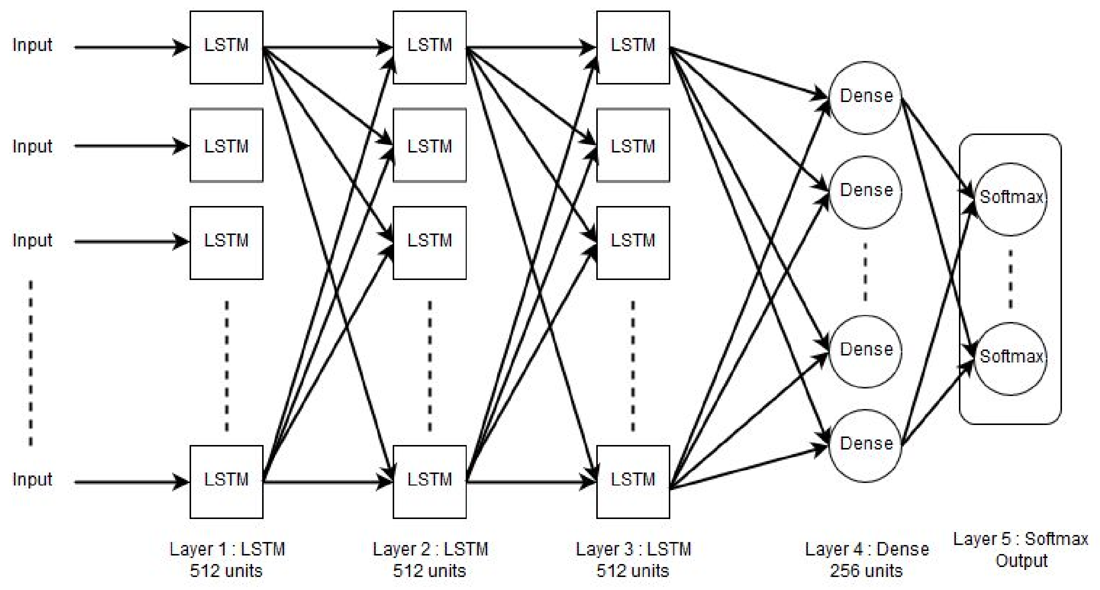


Fig. 6.2 Long Short Term Memory Architecture

The first model is a LSTM model, as shown in Fig. 2, is a sequential model which comprises of 3 layers of 512 LSTM units followed by a dense layer of 256 units. The output layer is a softmax layer having n_vocab units, where, n_vocab is the number of unique musical characters (notes and chords) in the input sample.

6.2.3 Low Level System Design (GAN)

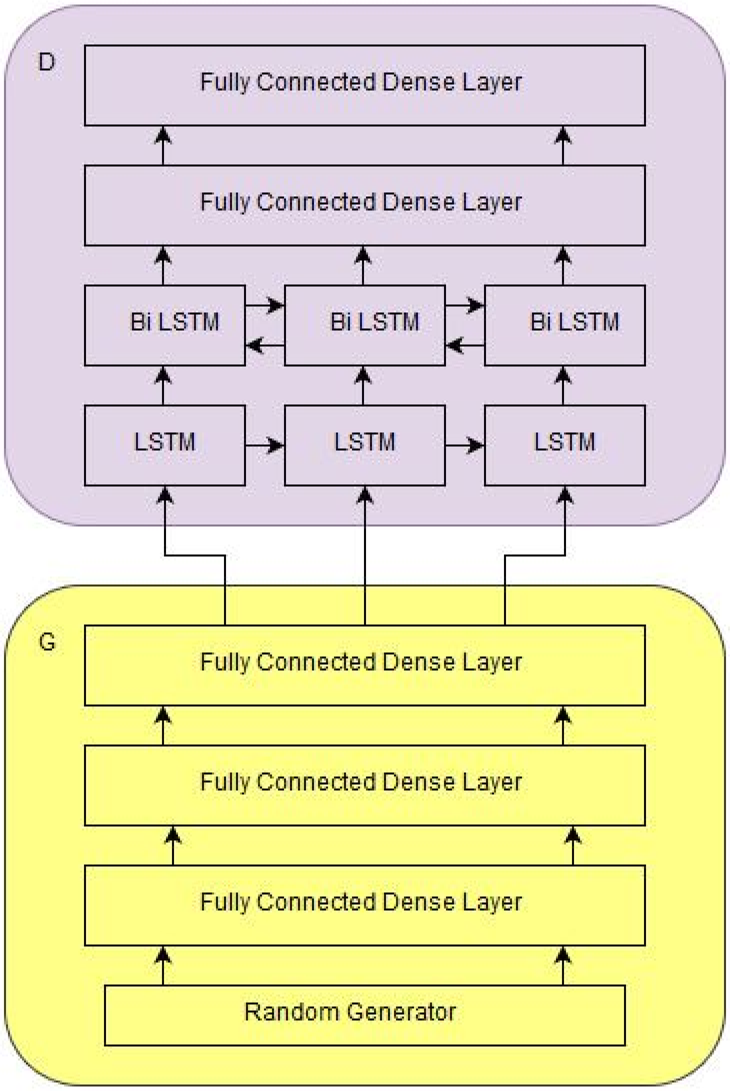

D
Fully Connected Dense Layer
Fully Connected Dense Layer
Bi LSTM
Bi LSTM
Bi LSTM
LSTM
LSTM
LSTM
G
Fully Connected Dense Layer
Fully Connected Dense Layer
Fully Connected Dense Layer
Random Generator

Fig. 6.3 GAN Model Architecture

The GAN Model, as shown in Fig. 3, has two stages, one generator and another discriminator. The generator network for the model has a CuDNNLSTM (faster implementation of LSTM) input layer with 512 units, five hidden layers which comprises of one CuDNNLSTM layer, two fully connected dense layers and two LeakyReLU (which allows a small, non-zero gradient when the unit is not active) layers. Output layer has a fully connected dense structure with sigmoid as activation function.

On the other hand the discriminator network has an input dense layer of 256 units. There are 8 hidden layers which consists of 3 LeakyReLU layers with alpha 0.2 which is the negative slope coefficient, 2 Dense layer and 3 Batch Normalization layers. Output layer has a fully connected dense network with tanh as activation function.

6.3 Graphical User Interface

//TODO

6.4 Sequence Diagram

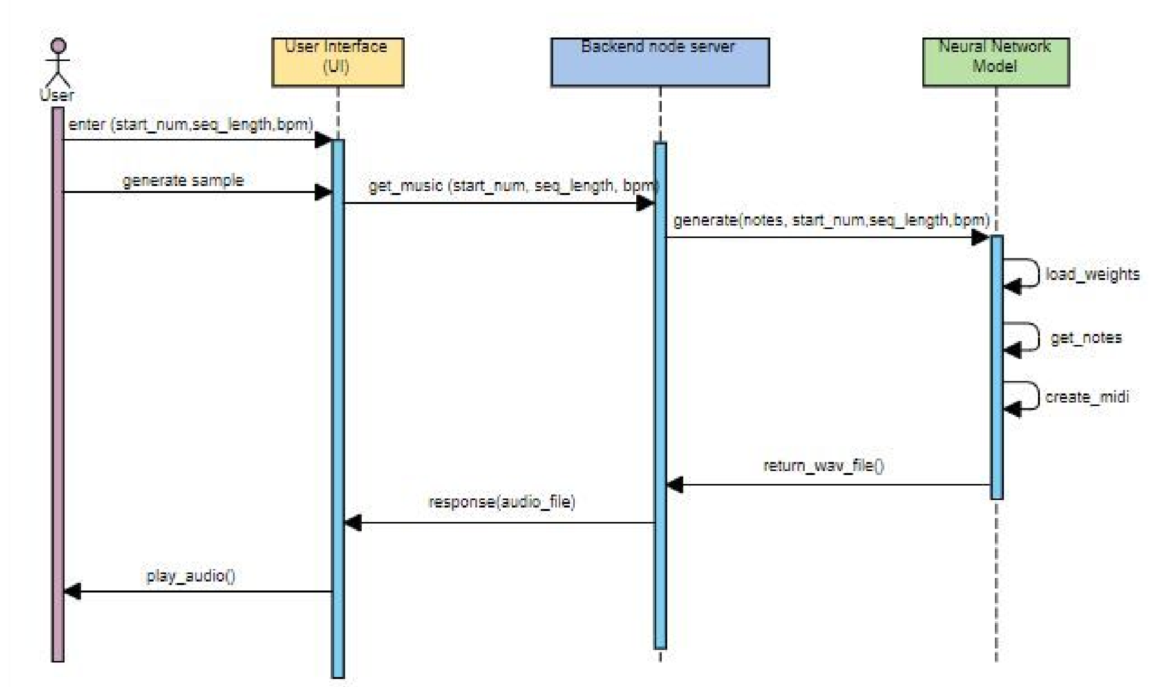


Fig. 6.4 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. In fig x.x we can see that we have four objects user, user interface, backend server, neural network model. The life cycle is started from the user who gives the necessary input to the UI and then calls to generate sample. UI takes the input from the user and then with the required parameters requests the backend server to get music. The backend server then calls the neural network model. The model then loads it's weights, then predicts a sequence of notes, these notes are then formatted to create a midi file. This file is then returned back to the backend server. The server then forwards the response to the UI, where the user plays the music.

6.5 Data flow diagram

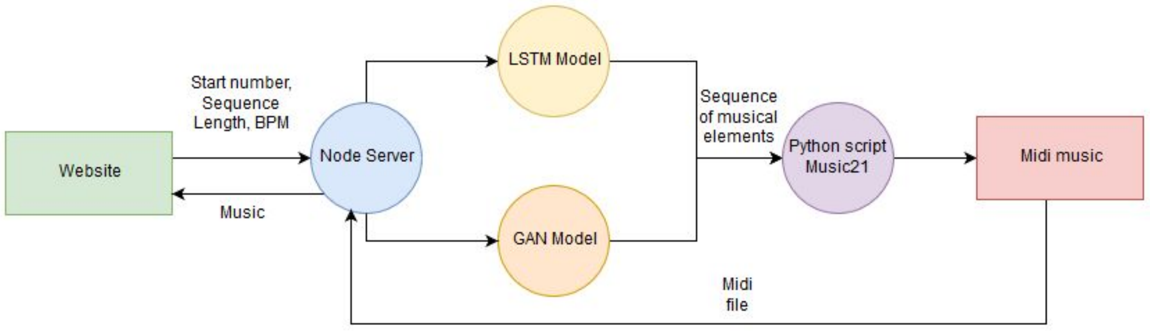


Fig. 6.5 Data Flow Diagram

In the figure x.x, it shows the main flow of the data in our project. We take the inputs from the user mainly start number which choose an input sequence before prediction, sequence length which gives length of the required music sample, BPM(Beats per minute) which represents the pace at which the notes are played. This data from the website front end is then sent to a server via a HTTP request. The server receives the request and then uses the above parameters to produce a musical sample. The output generated is just a sequence of musical elements which are then converted into an audible music format using music21. Once the generated music is in audible format it is then forwarded back to the website via HTTP request and hence can be played from the website frontend.

6.6 Conclusion

The design is implemented using sequence diagram, architectural design and data flow diagram. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are sometimes called event diagrams or event scenarios. An architectural design defines an

abstraction level at which the designers can specify the functional and performance behaviour of the system. A data flow diagram (DFD) is a graphical representation of the flow of data through an information system modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. All these diagrams help us in understanding the high level overview of the entire system.

Chapter 7

Implementation

7.1 Tools Introduction

Google Colab - is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. This helps machine learning and deep learning enthusiasts to be able to run code and access powerful computing resources, all for free from a browser.

Python 3.x - is an interpreted, high-level, general-purpose programming language. The language allows to leverage powerful deep learning techniques using various libraries.

Keras - Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow. It is designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It also allows use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

Music 21 - is a Python-based toolkit for computer-aided musicology. People use music21 to answer questions from musicology using computers, to study large datasets of music, to generate musical examples, to teach fundamentals of music theory, to edit musical notation, study music and the brain, and to compose music (both algorithmically and directly).

7.2 Technology Introduction

Our project primarily explores two Deep Learning techniques.

1) Long Short Term Memory (LSTM) - an artificial recurrent neural network (RNN) architecture, which processes single data points such as images, as well as entire sequences of data such as speech and video.

2) Generative adversarial network (GAN) - a class of machine learning systems, consisting of two neural networks contest with each other in a zero-sum game framework. It is a form of unsupervised learning. The generative network generates candidates while the discriminative network evaluates them

7.3 Explanation of Algorithm and how it is been implemented

7.3.1 LSTM

Long Short Term Memory (LSTM), shown in Fig 2., is an improvement over Recurrent Neural Network(RNN), introduced by Hochreiter and Schmidhuber in 1997, and improved upon to its current form in 1999 [18]. LSTMs are explicitly designed to solve vanishing gradient problem. Vanishing gradient is a problem uniquely faced by sequential data where an older input's influence over the output gradually reduces over long sequences of the input. LSTM solves this problem by using various gates and an activation function to maintain its state. The various inputs at time t are h_{t-1} (output from the last cell), x_t (current input) and it output h_t which will be the current output of the cell. C_t is the cell state at time t, and C_{t-1} is the cell state at time $t-1$. The various gated functions are

1. Forget gate: This is a function which helps us to decide what information to forget, it takes h_{t-1} and x_t as the inputs and performs the following function and outputs a number between 0 and 1, where 1 represents keep all the information and 0 represents forget everything.

   $f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$ ……………………………………………………..(7.1)

   where W_f, b_f are weights and bias for the equation respectively.

2. Input gate: This layer decides what new information is going to be stored in the cell state. $\hat{C}$ is calculated which is the new candidate for the cell state. It takes h_{t-1} and x_t as the inputs and produces C_t as the output.

   $i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$..(7.2)

   $\hat{C}_t = tanh (W_C \cdot [h_{t-1}, x_t] + b_c)$...(7.3)

   The cell state is then calculated as

   $C_t = f_t * C_{t-1} + i_t * \hat{C}_t$..(7.4)

   where W_i, b_i, W_c, b_c are weights and bias for the equations respectively.

3. Output gate: The final is produced based upon the cell state. It takes C_t, h_{t-1} and x_t as the input and produces h_t as the output.

   $o_t = \sigma (W_o \cdot [h_{t-1} , x_t] + b_o)$..(7.5)

   $h_t = o_t * \tanh(C_t)$..(7.6)

   where W_o, b_o are weights and bias for the equation respectively.

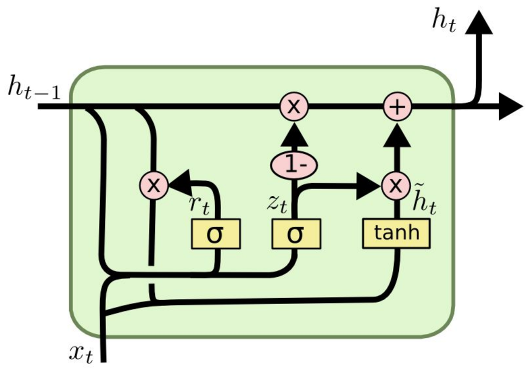


Fig. 7.1 LSTM Network

7.3.2 GAN

Generative Adversarial Network (GAN), shown in Fig 3., is an unsupervised deep learning algorithm proposed by Ian Goodfellow and few other researchers including

Yoshua Bengio in 2014 [2]. GAN comprises of two multi layered perceptrons nets called Generator and Discriminator. The Generator is pitted against an adversarial network called Discriminator. Generator’s task is to generate data that is very similar to training data and has to be indistinguishable from the real input data so that the discriminator can be tricked to identify it as real data. The task of the discriminator is to identify whether the data is real or fake. It uses two sets of input, one from the training dataset and the other from the one generated by the generator network for this purpose.

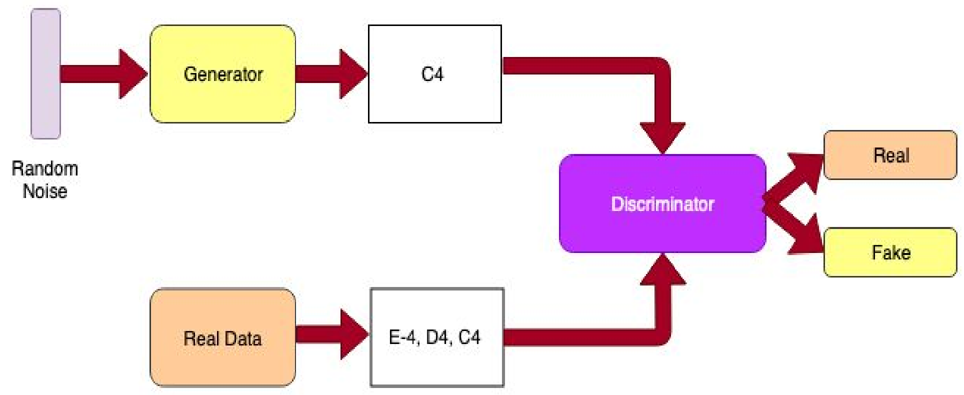


Fig. 7.2 Generative Adversarial Network

In the GAN model, the input to the generator G is a vector of random noises $z \in R^1$, and the output obtained by generator G is an h-by-w matrix $\widehat{X}$ = G(z) that seems to be real to discriminator D. GAN learns G and D by using the following formula

$$min_{G}\ max_{D}\ V(D,G) = E_{X\sim pdata(X)}[log(D(X))] + E_{z\sim pz(z)}[log(1 - D(G(z)))] \ldots\ldots\ldots\ldots(7.7)$$

where X ~ pdata(X) denotes the operation of sampling from real music data, and z ~ pz(z) the sampling from a random distribution.

7.3.3 Data Preprocessing

The dataset containing 70 midi files are first loaded into Music21 object stream and parsed using Music21's parse module. This stream object lists all the notes and chords in all the midi files. A sequential list of notes and chords will act as input for out model. These notes and chords are mapped or encoded from string categorical data to numerical data. A sequence length of 100 notes/chords was used as input sequences to the model which helps to predict the next note/chord that will come after the input sequence. Thus, the whole sequential list is divided into a number of chunks each of length 100 which will be fed to the LSTM model. Final step in data preprocessing is to normalise the input sequences. Once normalization is done the model is ready to train.

7.3.4. Training

Dataset: The dataset used was a subsample of the MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) Dataset. The entire dataset consists of over 172 hours of virtuosic piano performances captured with fine alignment (~3 ms) between note labels and audio waveforms.

This project used a part of the 2017 MAESTRO dataset, which consisted of over 70 songs of various piano compositions of over a few hours.

Dropout: is a regularization technique used to overcome overfitting by using a fraction of input units. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

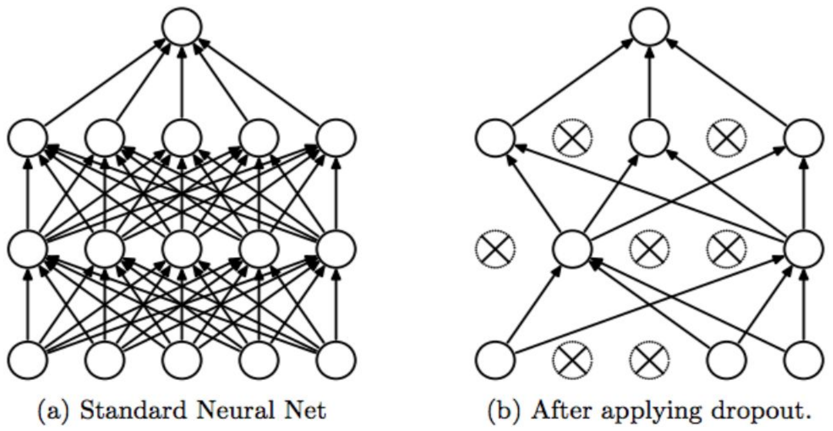


Fig. 7.3 Dropout

Loss function : It's a method of evaluating how well specific algorithm models the given data. If predictions deviates too much from actual results, loss function outputs a high number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction.

Categorical Cross Entropy - Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

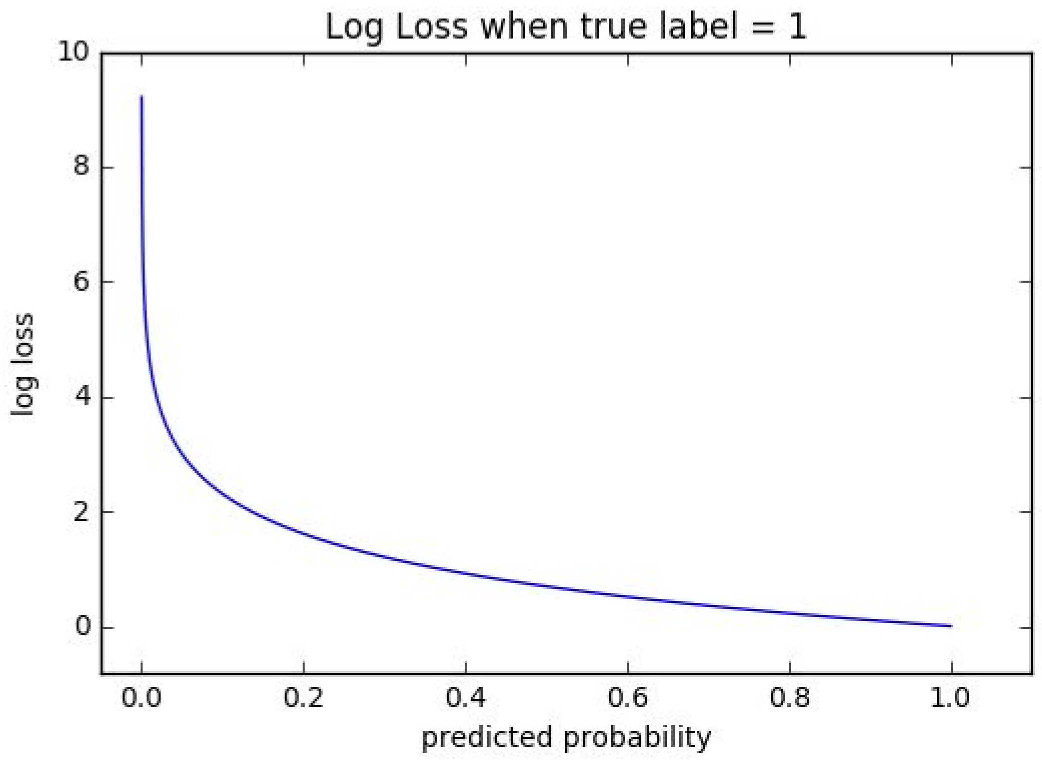


Fig. 7.4 Categorical Cross Entropy

Activation functions :

Sigmoid - A sigmoid function is a mathematical function having a characteristic "S" - shaped curve or sigmoid curve. Often, sigmoid function refers to the special case of the logistic function and defined by the formula. A graph for sigmoid function is shown below in fig x.

$f(x) = \frac{1}{1+e^{-x}}$ ……………………………………………………………………..(7.8)

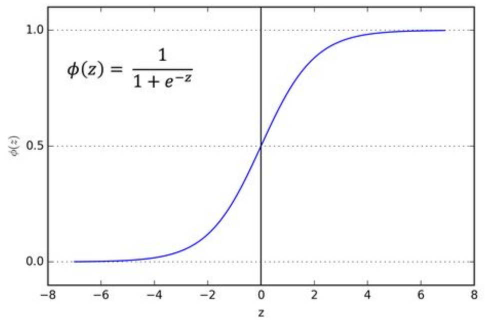


Fig. 7.5 Sigmoid Curve

Tanh - The Hyperbolic Tangent activation function, like the logistic sigmoid, is also an 's' shaped curve, but instead outputs values between -1 and 1.

$$y = S(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \text{(7.9)}$$

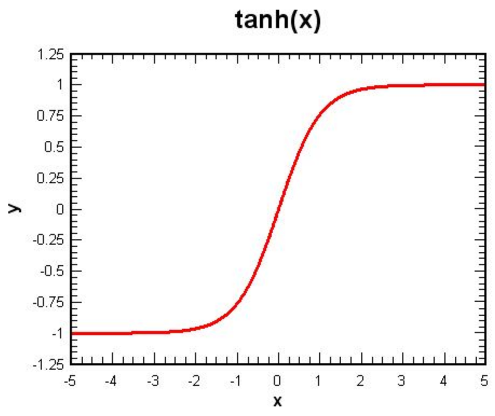


Fig. 7.6 Hyperbolic Tangent Curve

Softmax - Softmax function calculates the probabilities distribution of the event over 'n' different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. The main advantage of using Softmax is the output probabilities range. The range will 0 to 1, and the sum of all the probabilities will be equal to one. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)} \quad \text{...}(7.10)$$

The LSTM model architecture has *3* LSTM layers which takes musical sequences as input and returns sequences or matrix as output. We use Dropout, which is a regularization technique used to overcome overfitting by using a fraction of input units. The *x* Dense Layers where each input unit is connected with output unit. The Activation Layer uses softmax as activation function to produce the final output of the network. Categorical Cross Entropy is used to calculate loss for each iteration because each output belongs to a single class, and this loss is optimized using Adam optimizer.

The model was then trained for 200 epochs and weights were saved after every iteration where loss is lesser than the all the previous iteration.

7.3.5 Generation of Music

For output the model requires three parameters, a start number which takes any sequence from pre generated sequence during training, as the start sequence. Sequence length which specifies how long the generated sample will be and beats per minute which is sets the tempo of the sample. The model uses these three inputs to generate a sequence of encoded notes and chords, which are decoded and added to a MIDI stream for MIDI file generation. The elements are temporally spaced out with a fixed offset of 0.5 from each other. The midi file thus generated is then converted to .wav file for easier playing.

7.3.6 GAN

In the GAN model, the input to the generator G is a vector of random noises z $\in$ R^1 , and the output obtained by generator G is an h-by-w matrix $\widehat{X}$ = G(z) that seems to be real to discriminator D. GAN learns G and D by using the following formula

$$min_G \, max_D \, V(D, G) = E_{X\text{~}pdata(X)} [log(D(X))] + E_{z\text{~}pz(z)} [log(1 - D(G(z)))] \text{...........}(7.11)$$

where X ~ pdata(X) denotes the operation of sampling from real music data, and z ~ pz(z) the sampling from a random distribution.

This GAN model was trained for 5000 epochs with a learning rate of 0.0002 and loss obtained by binary cross entropy was optimized using Adam Optimizer.

The discriminator is trained to distinguish between real music data and generated data and the goal of generator is to generate music that is very close to the real music that is feeded as the input data so that the discriminator identifies the generated one as the real music. Hence, serving the purpose of generating good quality music which is as good as the dataset being used.

7.5 Information about the implementation of Modules

Generate - A wrapper function which takes a random number denoting an input sample, length of output sequence, and the bpm of output, as input parameters. It calls the subroutines that prepare sequences, creates the deep learning model, generate music and create the midi file.

Get Notes - A subroutine that parses all the midi files in the training dataset directory, extracts the musical elements, namely- notes and chords, and returns an encoded sequential list of notes and chords as shown in Fig x.x

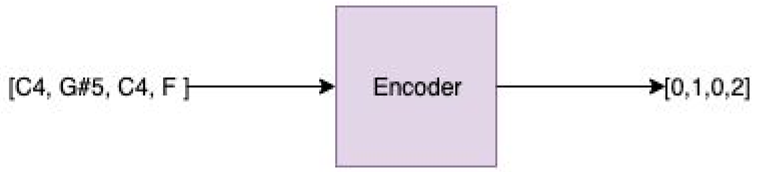
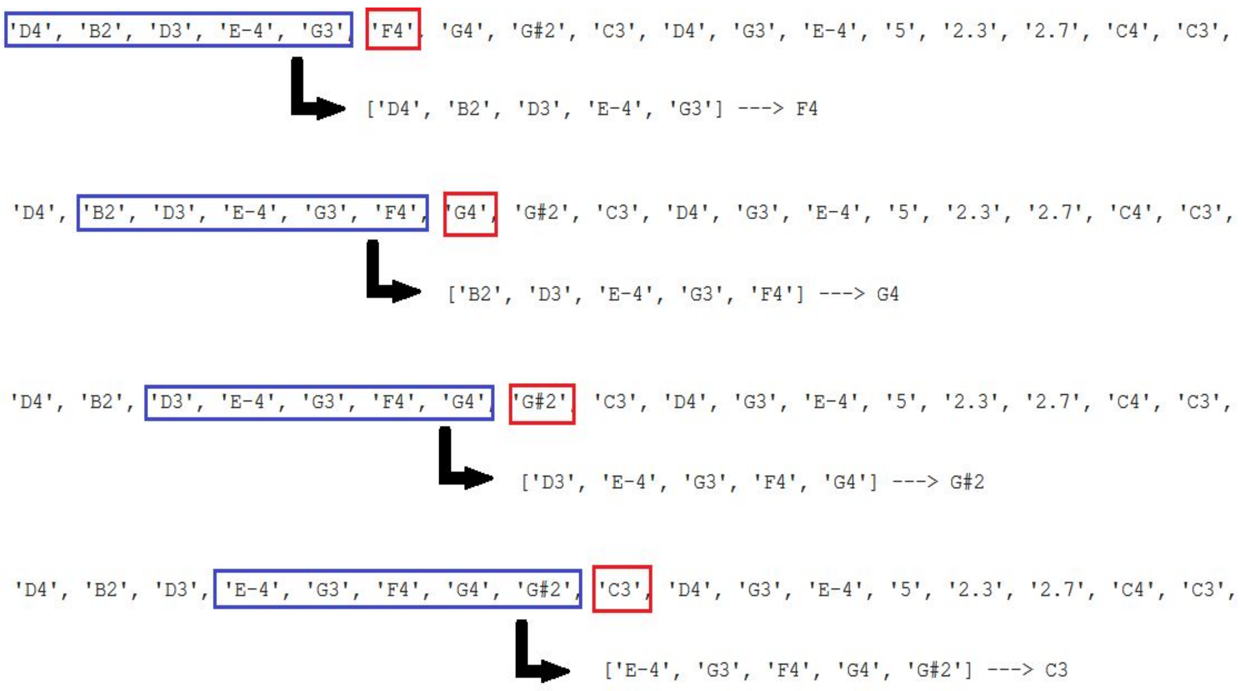


Fig. 7.7 Categorical to Numerical encoder

Prepare Sequences - A subroutine which takes the entire list of musical elements in the training set, and the number of unique occurences of these musical elements in the vocabulary. The subroutine prepares the input sequences for training.


```
'D4', 'B2', 'D3', 'E-4', 'G3', 'F4', 'G4', 'G#2', 'C3', 'D4', 'G3', 'E-4', '5', '2.3', '2.7', 'C4', 'C3',
                      ['D4', 'B2', 'D3', 'E-4', 'G3'] ---> F4

'D4', 'B2', 'D3', 'E-4', 'G3', 'F4', 'G4', 'G#2', 'C3', 'D4', 'G3', 'E-4', '5', '2.3', '2.7', 'C4', 'C3',
                           ['B2', 'D3', 'E-4', 'G3', 'F4'] ---> G4

'D4', 'B2', 'D3', 'E-4', 'G3', 'F4', 'G4', 'G#2', 'C3', 'D4', 'G3', 'E-4', '5', '2.3', '2.7', 'C4', 'C3',
                                ['D3', 'E-4', 'G3', 'F4', 'G4'] ---> G#2

'D4', 'B2', 'D3', 'E-4', 'G3', 'F4', 'G4', 'G#2', 'C3', 'D4', 'G3', 'E-4', '5', '2.3', '2.7', 'C4', 'C3',
                                     ['E-4', 'G3', 'F4', 'G4', 'G#2'] ---> C3
```

Fig. 7.8 Sample preparation by shifting the moving window by 1 step. Blue denotes the input sequence part of the sample, red denotes the next element to be predicted, i.e. output.

Create Network - This subroutine takes two parameters, one which is a matrix which acts as the input to the neural network and the other being the number of unique occurences of the musical elements in the vocabulary. This module defines the network architecture consisting of various Bidirectional LSTM layers. It also consists of Dropout and Dense layers. The model uses Softmax as its activation function.

Generate Notes - This is a subroutine which takes a number of parameters to be able to generate the output sequence of the model. It takes in the compiled model with the model weights, the sequential list of notes and chords, the normalized matrix which acted as the input to the network, the unique occurences of the musical elements in the vocabulary, the start number which defines the start sequence of the sample, and the sequence length describes the duration of the song.

Create MIDI - This module takes two parameters, the first one is the predicted output sequence from the model and the second one is beats per minute which sets the tempo of the sample generated. It uses Music21 library to parse the sequences of notes and chords back into a midi file which is returned to the calling module.

GAN class - The GAN class consists of methods for instantiating and initialising the generator and discriminator models, training the models and generate music.

7.6. Conclusion

The two models - LSTM and GAN were designed as per specified in the Low Level System Design(Fig x. and Fig x.), were trained on the training sequences using the algorithms explained above,

Chapter 8

Testing

8.1 Introduction

After the model is trained, the next phase is testing of the model.The testing phase involves analysing the accuracy and loss of the deep learning model, as well as quality check of the music generated by the model.

8.2 Testing Tools and Environment

The tools and environment used for testing are:

- Anaconda
- Google Colab Notebook
- Jupyter Notebook
- Garage band / MIDI players

8.3 Test cases

Duration of sample : The length of the sample was given as an input to the user. The model generated samples of the specific sequence length given by the user.

Beats per minute (BPM) : The tempo of a musical piece is key to how good it sounds. The user specified the BPM for the sample to be generated and the model generated the sample for corresponding BPM value.

Generation of different sounding samples : The model was trained on a large dataset so as to produce different sounding samples. Multiple samples were generated randomly by the model and the samples sounded unique enough to the user.

Survey of samples : A survey was conducted, wherein 7 musical samples were to be rated on a quality scale of 1-5, and whether the sample was composed by a human or AI.

Chapter 9

Results

9.1 Result Snapshots

The Snapshots of the results obtained are shown below.

a) Website Dashboard

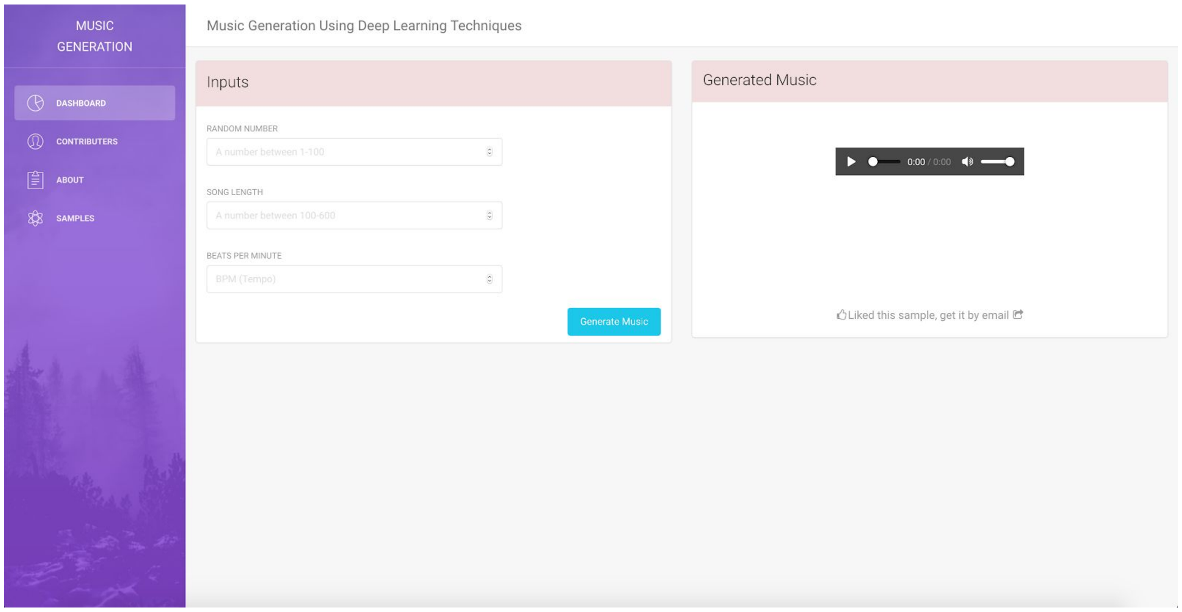
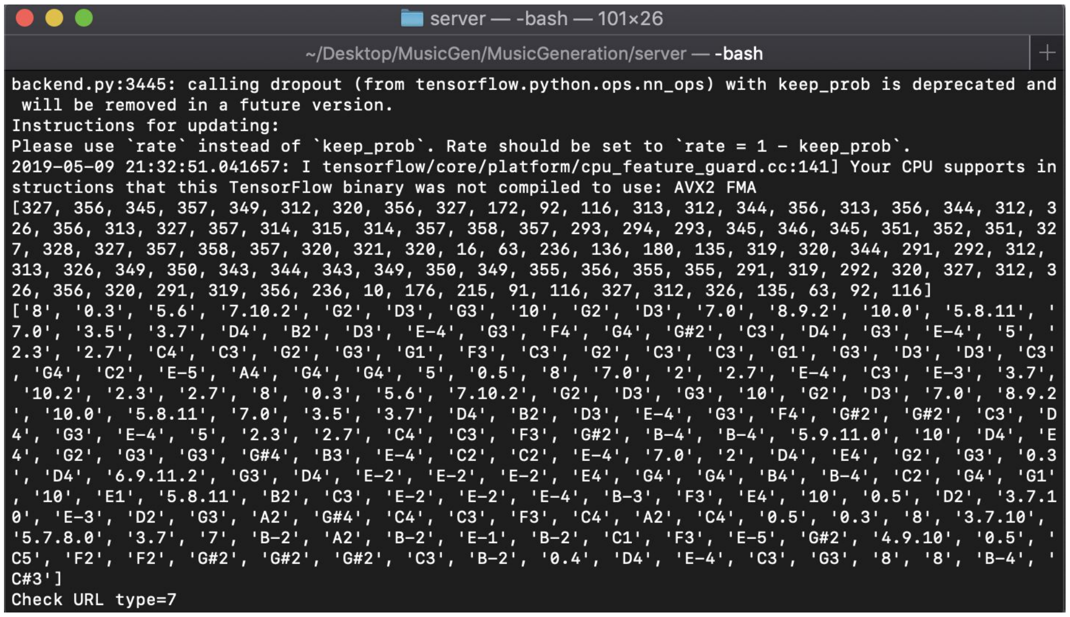


Fig. 9.1 Website dashboard

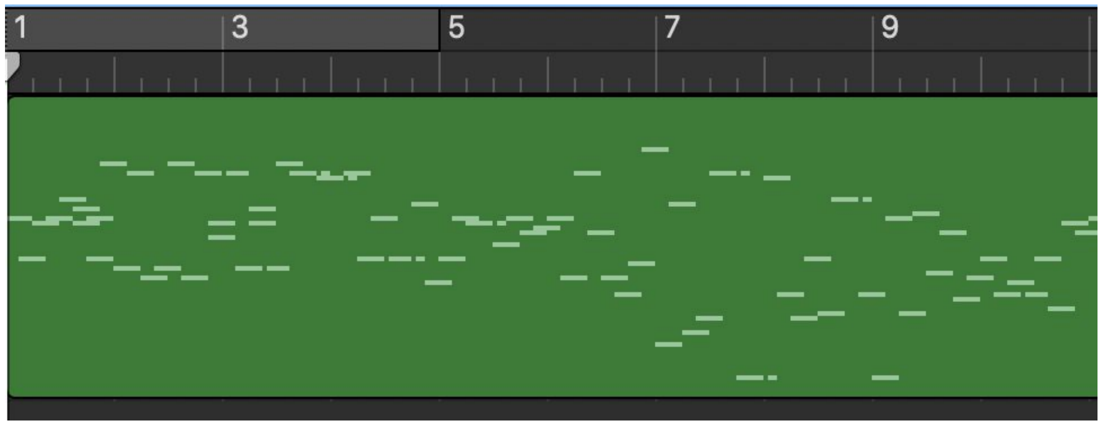
b) Notes and chords generated



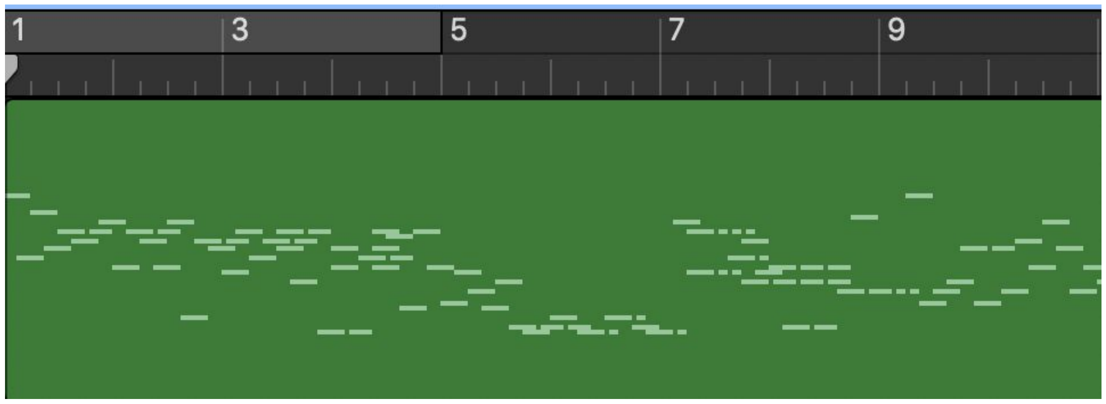
```
backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and
 will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
2019-05-09 21:32:51.041657: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports in
structions that this TensorFlow binary was not compiled to use: AVX2 FMA
[327, 356, 345, 357, 349, 312, 320, 356, 327, 172, 92, 116, 313, 312, 344, 356, 313, 356, 344, 312, 3
26, 356, 313, 327, 357, 314, 315, 314, 357, 358, 357, 293, 294, 293, 345, 346, 345, 351, 352, 351, 32
7, 328, 327, 357, 358, 357, 320, 321, 320, 16, 63, 236, 136, 180, 135, 319, 320, 344, 291, 292, 312,
313, 326, 349, 350, 343, 344, 343, 349, 350, 349, 355, 356, 355, 355, 291, 319, 292, 320, 327, 312, 3
26, 356, 320, 291, 319, 356, 236, 10, 176, 215, 91, 116, 327, 312, 326, 135, 63, 92, 116]
['8', '0.3', '5.6', '7.10.2', 'G2', 'D3', 'G3', '10', 'G2', 'D3', '7.0', '8.9.2', '10.0', '5.8.11', '
7.0', '3.5', '3.7', 'D4', 'B2', 'D3', 'E-4', 'G3', 'F4', 'G4', 'G#2', 'C3', 'D4', 'G3', 'E-4', '5', '
2.3', '2.7', 'C4', 'C3', 'G2', 'G3', 'G1', 'F3', 'C3', 'G2', 'C3', 'C3', 'G1', 'G3', 'D3', 'D3', 'C3'
, 'G4', 'C2', 'E-5', 'A4', 'G4', 'G4', '5', '0.5', '8', '7.0', '2', '2.7', 'E-4', 'C3', 'E-3', '3.7',
 '10.2', '2.3', '2.7', '8', '0.3', '5.6', '7.10.2', 'G2', 'D3', 'G3', '10', 'G2', 'D3', '7.0', '8.9.2
', '10.0', '5.8.11', '7.0', '3.5', '3.7', 'D4', 'B2', 'D3', 'E-4', 'G3', 'F4', 'G#2', 'G#2', 'C3', 'D
4', 'G3', 'E-4', '5', '2.3', '2.7', 'C4', 'C3', 'F3', 'G#2', 'B-4', 'B-4', '5.9.11.0', '10', 'D4', 'E
4', 'G2', 'G3', 'G3', 'G#4', 'B3', 'E-4', 'C2', 'C2', 'E-4', '7.0', '2', 'D4', 'E4', 'G2', 'G3', '0.3
', 'D4', '6.9.11.2', 'G3', 'D4', 'E-2', 'E-2', 'E-2', 'E4', 'G4', 'G4', 'B4', 'B-4', 'C2', 'G4', 'G1'
, '10', 'E1', '5.8.11', 'B2', 'C3', 'E-2', 'E-2', 'E-4', 'B-3', 'F3', 'E4', '10', '0.5', 'D2', '3.7.1
0', 'E-3', 'D2', 'G3', 'A2', 'G#4', 'C4', 'C3', 'F3', 'C4', 'A2', 'C4', '0.5', '0.3', '8', '3.7.10',
'5.7.8.0', '3.7', '7', 'B-2', 'A2', 'B-2', 'E-1', 'B-2', 'C1', 'F3', 'E-5', 'G#2', '4.9.10', '0.5', '
C5', 'F2', 'F2', 'G#2', 'G#2', 'G#2', 'C3', 'B-2', '0.4', 'D4', 'E-4', 'C3', 'G3', '8', '8', 'B-4', '
C#3']
Check URL type=7
```

Fig. 9.2 Terminal output of generation of notes and chords

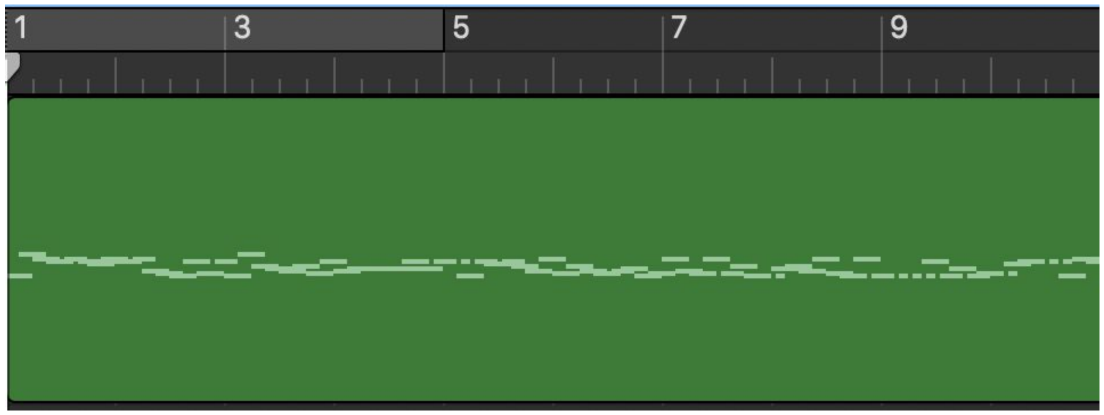
c) Musical samples



(a)



(b)



(c)

Fig. 9.3 Visual representation for the musical samples generated by a) LSTM sample 1, b) LSTM sample 2, c) GAN sample 1

9.2 Comparison results tables

The models generated music of specified lengths and BPM (Beats per minute). To be able to evaluate and compare the LSTM and the GAN models, we look at the loss values when the respective model was trained in Fig. 9.2.1 and Fig. 9.2.2..

The loss value for the LSTM model, was 0.8343. The GAN model's loss was about 0.8901. This shows that the LSTM model quantitatively performs slightly better than the GAN model. But, the loss values only aren't enough to determine model performances, especially, because of the subjectivity attributed to the musical samples. Thus, a survey was also conducted to establish the results.

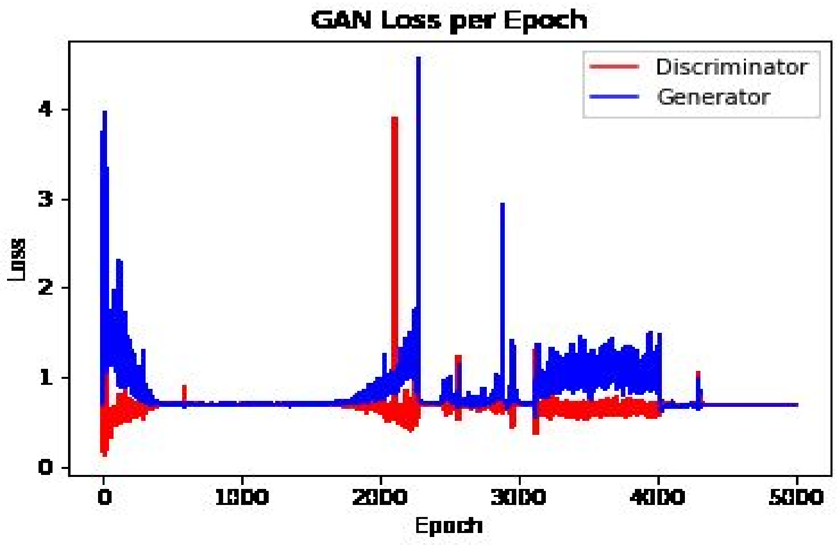


Fig. 9.4 GAN Loss per Epoch

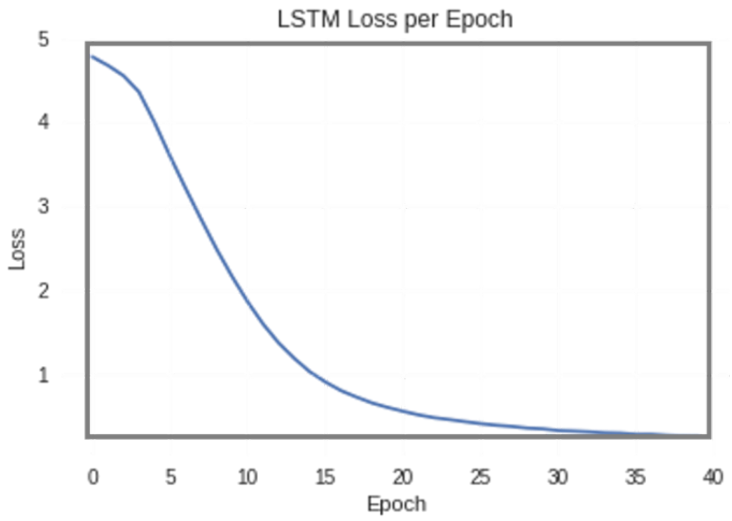


Fig. 9.5 LSTM Loss per Epoch

9.3 Performance analysis

A survey was conducted to evaluate how good the samples generated by the models were. Getting a diverse set of peers to opine on the music samples generated helped in evaluating the quality of the model, i.e. the participants in the survey varied from people who were professional/amateur musicians with some knowledge of music theory to people who have absolutely no knowledge of music. The responses of more than 100 participants were taken into account for judging the quality of music produced by the model proposed. The charts displayed below, show the results of the survey. The survey expected participants to primarily answer to questions for each sample. They were asked to rate the sample in the following categories on a scale of 1 to 5 - very poor, poor, fair, good or excellent, i.e. 1 being very poor and 5 being excellent. The participants were also asked to classify the samples as human generated or AI generated.

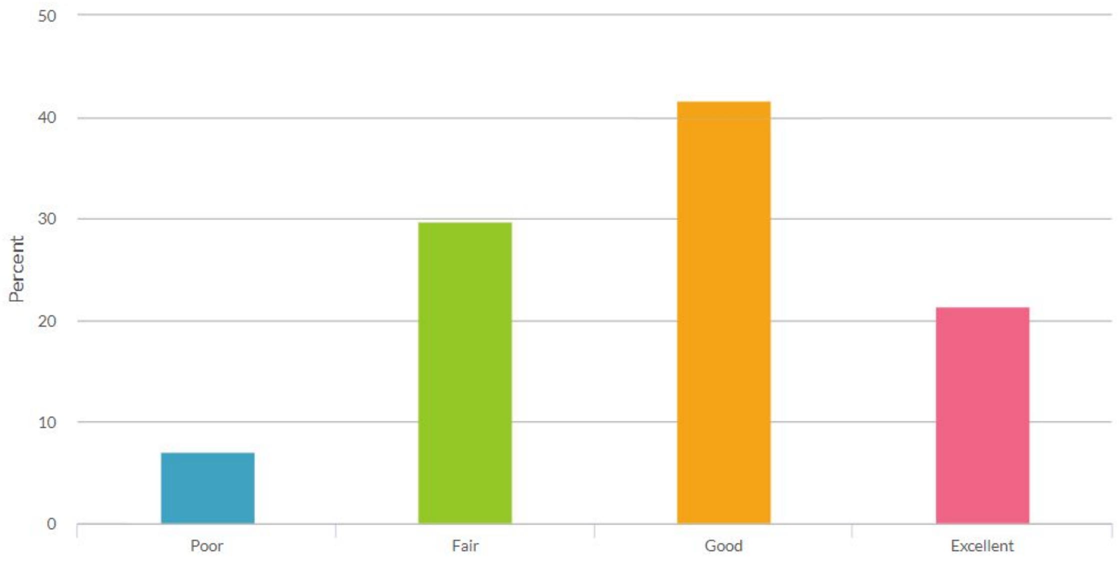


Fig. 9.6 Bar graph of responses of Music sample 1 - LSTM

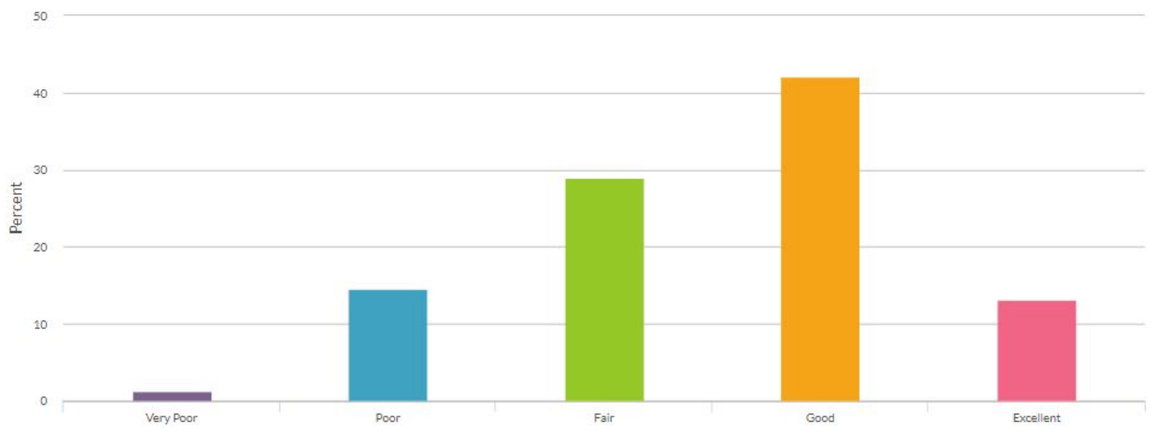


Fig. 9.7 Bar graph of responses of Music sample 2 - GAN

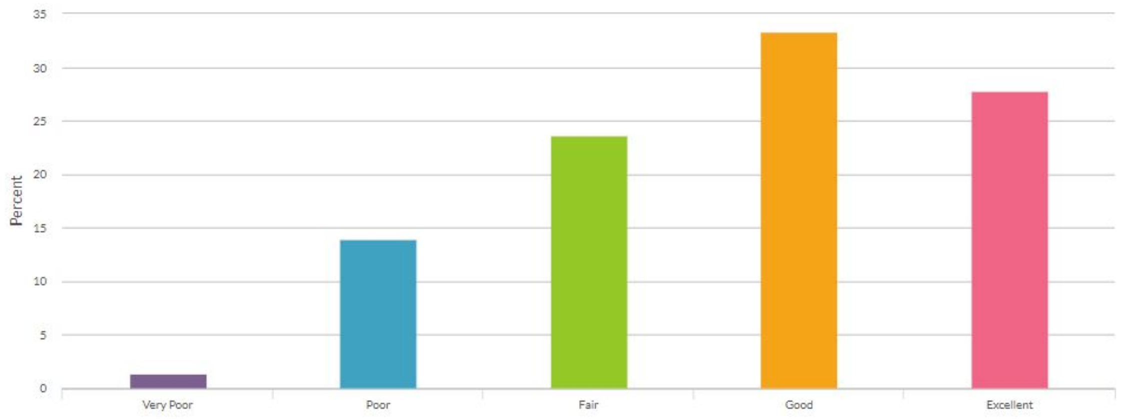


Fig. 9.8 Bar graph of responses of Music sample 3 - LSTM

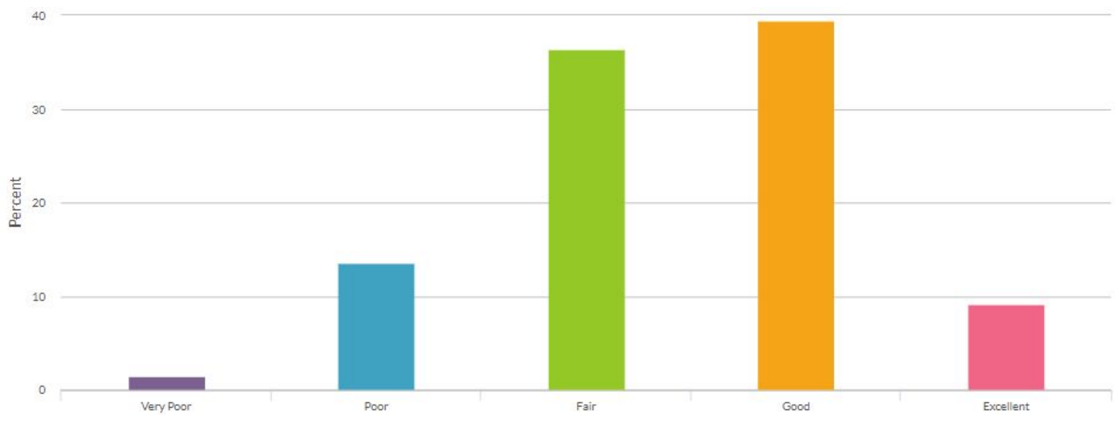


Fig. 9.9 Bar graph of responses of Music sample 4 - GAN

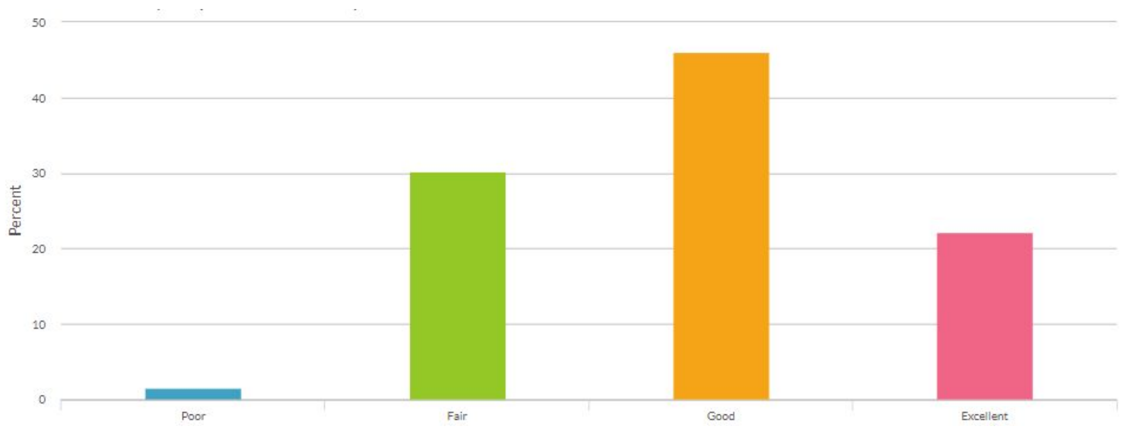


Fig. 9.10 Bar graph of responses of Music sample 5 - LSTM

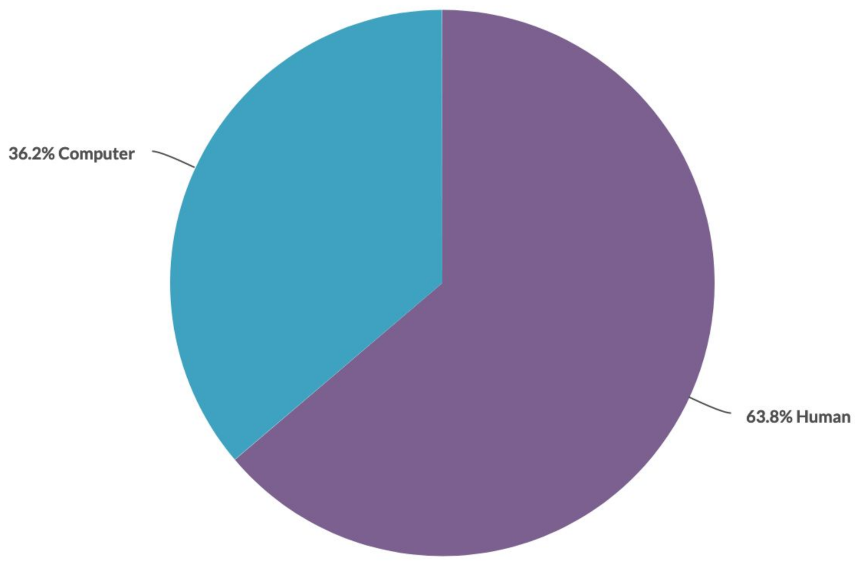


Fig. 9.11 Pie Chart of responses for classification of human or AI generated

From the figures Fig 9.6, Fig 9.7, Fig 9.8, Fig 9.9, and Fig 9.10 generated from the responses obtained from the survey, the following could be inferred :

- On an average 29.8% of the participants rated the music generated by both the models as Fair, 39.8% as Good, and 18% as Excellent. This means
- Second, only one of the LSTM generated samples in the survey even had responses in the very poor section. The GAN samples had an average of 16% rated as poor or very poor, whereas the LSTM samples had an average of 9%.

- Also, it was observed that the LSTM generated samples on an average had 63.6% scores of good and excellent as opposed to the GAN generated samples which had an average of 52%

Thus, the models are able to produce good quality music. Further, the LSTM model's samples were received much better than the samples generated by the GAN model.

Fig. 9.11 shows the results of the classification of LSTM generated samples into Human generated and AI generated. The pie chart shows that 63.8% of participants classified the LSTM samples as human generated. This suggests that the music was nuanced and pleasing enough to be identified as real music generated by a person, and thus reflects the model's capability to learn the musical patterns well.

Chapter 10

Conclusion and scope for future work

With the use of deep learning techniques, the model proposed is able to generate human like music. The two primary models explored were LSTMs and GANs. Using both the loss value plots and the survey conducted, we observe that the LSTM model performs better than our GAN model. The LSTM model is able to learn the patterns in the sequence of music as it helps preserve the error that can be back-propagated through time and layers.

The model can be improved in the future by incorporating multi instruments. This would mean that the MIDI files that are used would have various components for each instrument and the network should be robust enough to be able to parse the individual components appropriately. Harmonising with multiple instruments at different parts of the sample would make the music generated sound better.

Chapter 11

References

1. SampleRNN: An Unconditional end-to-end Neural Audio Generation Model
2. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014
3. Generating Sequences With Recurrent Neural Networks, Alex Graves
4. Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks, Douglas Eck and JÄurgen Schmidhuber IDSIA Istituto Dalle Molle di Studi sull'Intelligenza Arti¯ciale Galleria 2, 6928 Manno, Switzerland.
5. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription - Nicolas Boulanger-Lewandowski, Yoshua Bengio, Pascal Vincent
6. Music Generation from Statistical Models - Darrell Conklin
7. Character-aware neural language models - Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush
8. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, Yoshua Bengio
9. Deep Learning for Music - Allen Huang, Raymond Wu
10. A First Look at Music Composition using LSTM Recurrent Neural Networks - Douglas Eck, Jurgen Schmidhuber
11. Text-based LSTM networks for Automatic Music Composition - Keunwoo Choi, George Fazekas, Mark Sandler
12. Modeling Temporal Dependencies in High-Dimensional Sequences:Application to Polyphonic Music Generation and Transcription - Nicolas Boulanger-Lewandowski, Yoshua Bengio, Pascal Vincent

13. Composing Music with BPTT and LSTM Networks: Comparing Learning and Generalization Aspects - Débora C Corrêa, José HSaito, Sandra Abib
14. DeepJ: Style-Specific Music Generation - Huanru Henry Mao, Taylor Shin, Garrison W Cittrell
15. JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs - Gino Brunner, Yuyi Wang, Roger Wattenhofer and Jonas Wiesendanger
16. Generating musical expression of MIDI music with LSTM neural network - Maria, Adrian, Bartlomiej