A PROJECT REPORT

ON

"MUSIC GENERATION WITH A.I"

Submitted to

UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER ENGINEERING

BY

DEEPAK PRAJAPATI 17CO22
ASHRAF SONDE 17CO51
SUFYAN SHAHA 17CO52

UNDER THE GUIDANCE OF
PROF. IRFAN JAMKHANDIKAR

DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam's Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

2020-2021

AFFILIATED TO

UNIVERSITY OF MUMBAI

A PROJECT II REPORT
ON

"MUSIC GENERATION WITH A.I"

Submitted to
UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER ENGINEERING

BY

DEEPAK PRAJAPATI 17CO22
ASHRAF SONDE 17CO51
SUFYAN SHAHA 17CO52

UNDER THE GUIDANCE OF
PROF. IRFAN JAMKHANDIKAR

DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam's Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

2020-2021
AFFILIATED TO

UNIVERSITY OF MUMBAI

Anjuman-i-Islam's Kalsekar Technical Campus

Department of Computer Engineering

SCHOOL OF ENGINEERING & TECHNOLOGY

Plot No. 2 3, Sector - 16, Near Thana Naka,

Khandagaon, New Panvel - 410206

CERTIFICATE

This is certify that the project entitled

"MUSIC GENERATION WITH A.I"

submitted by

DEEPAK PRAJAPATI 17CO22
ASHRAF SONDE 17CO51
SUFYAN SHAHA 17CO52

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Engineering) at ***Anjuman-I-Islam's Kalsekar Technical Campus, Navi Mumbai*** under the University of MUMBAI. This work is done during year 2020-2021, under our guidance.

Date: 25 / 0 5 / 2021

(Prof. IRFAN JAMKHANDIKAR)
Project Supervisor

(Prof. KALPANA BODKE)
Project Coordinator

(Prof. TABREZ KHAN)
HOD, Computer Department

DR. ABDUL RAZAK HONNUTAGI
Director

External Examiner

Acknowledgements

I would like to take the opportunity to express my sincere thanks to my guide **Irfan Jamkhandikar**, Assistant Professor, Department of Computer Engineering, AIKTC, School of Engineering, Panvel for his invaluable support and guidance throughout my project research work. Without his kind guidance & support this was not possible.

I am grateful to him/her for his timely feedback which helped me track and schedule the process effectively. His time, ideas and encouragement that he gave is help me to complete my project efficiently.

We would like to express deepest appreciation towards **DR. ABDUL RAZAK HONNUTAGI**, Director, AIKTC, Navi Mumbai, **Prof. TABREZ KHAN**, Head of Department of Computer Engineering and **Prof. KALPANA BODKE**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

DEEPAK PRAJAPATI

ASHRAF SONDE

SUFYAN SHAHA

Project I Approval for Bachelor of Engineering

This project entitled "**MUSIC GENERATION WITH A.I**" by ***Deepak Prajapati, Ashraf Sonde, Sufyan Shaha*** is approved for the degree of

Bachelor of Engineering in Department of Computer Engineering.

Examiners

1. ....
2. ....

Supervisors

1. ....
2. ....

Chairman

....

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Student Name: Deepak Prajapati
Roll Number: 17CO22
Student Name: Ashraf Sonde
Roll Number: 17CO51
Student Name: Sufyan Shaha
Roll Number: 17CO52

ABSTRACT

Title: Music generation with Artificial Intelligence

Today in the world of growing technology the domain of artificial intelligence is the pioneer. There are majority of the advancements and applications of Artificial Intelligence that we hear about refer to a category of algorithms known as Machine Learning. Self-learning algorithms use statistics to draw models from huge amounts of data. Machine learning is able to make very precise assumptions about what we do, about the next activity we might want to do. Alongside visual art and creative writing, musical composition is another core act of creativity that we consider to be uniquely human. We will create a model that will generate completely new music.

Keywords :
Machine Learning, Training set, Training Data, Deep learning, Data pre-processing, Artificial Intelligence, Discriminator, Generator, Attention.

Abbreviations :

GAN – Generative Adversarial Network
A.I – Artificial Intelligence

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Generating music has a few notable differences from generating images and videos. First, music is an art of time, necessitating a temporal model. Second, music is usually composed of multiple instruments/tracks with their own temporal dynamics, but collectively they unfold over time interdependently. Lastly, musical notes are often grouped into chords, arpeggios or melodies in polyphonic music, and thereby introducing a chronological ordering of notes is not naturally suitable

1.1 Purpose

Using AI as a tool to make music or aid musicians. Composing a new music will become easier. The objective of this project is to generate music using Neural Networks. The model will be predicting monophonic music notes to generate new music.

1.2 Project Scope

The basic idea is to generate music using A.I or more specifically by using neural networks. We will use attention mechanism to generate future music notes. Using attention mechanism, our model will be able to choose which previous notes to focus on in order to predict which notes will appear next. Artificial intelligence is also helping the industry with A&R (artist and repertoire) discovery. It's always been challenging to comb through music and find promising artists that haven't signed to a label, but it's even more overwhelming with the deluge of streaming music today. Warner Music Group acquired a tech start-up last year that uses an algorithm to review social, streaming and touring data to find promising talent. Apple also acquired a start-up that specializes in music analytics to support the A&R process. AI is behind the scenes influencing the music we listen to in many ways.

1.3 Project Goals and Objectives

Using AI as a tool to make music or aid musicians. Composing a new music will become easier. The objective of this project is to generate music using Neural Networks. The model will be predicting monophonic music notes to generate new music.

1.4 Organization of Report

Title

Summary

List of contents

Introduction

Main body of the report

Conclusions

References

Chapter 2

Literature Survey

2.1 MuseGAN: Multi-track Sequential Generative Adversarial networks

It shows that the models can generate coherent music from scratch. Given a specific track composed by human, it can generate additional tracks to accompany it

2.1.1 Advantages of Paper

a. New musical notes can be generated from scratch.

b. Track-conditional Generation-Assumes that the bar sequence of one specific track is given and learn the temporal structure to generate the remaining tracks.

2.1.2 Disadvantages of Paper

a. Because of the injected noise and imperfect elementwise measures such as the squared error, the generated samples are often blurry.
b. Non-convergence: the model parameters oscillate, destabilize and never converge.

2.1.3 How to overcome the problems mentioned in Paper

a. We will capture previous musical notes with attention mechanism.
b. We will capture previous musical notes with attention mechanism. An attention model is a method that takes n arguments y1, yn (in the preceding examples, the yi would be the hi), and a context c. It return a vector z which is supposed to be the "summary" of the yi, focusing on information linked to the context c.

2.2 Conditional LSTM-GAN for Melody Generation from Lyrics.

Melody generation from lyrics, which contains a deep LSTM generator and a deep LSTM discriminator. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, music generation and more.

2.2.1 Advantages of Paper

a. Language modelling or text generation, that involves the computation of words when a sequence of words is fed as input. Language models can be operated at the character level, n-gram level, sentence level or even paragraph level.

b. Music generation which is quite similar to that of text generation where LSTMs predict musical notes instead of text by analyzing a combination of given notes fed as input.

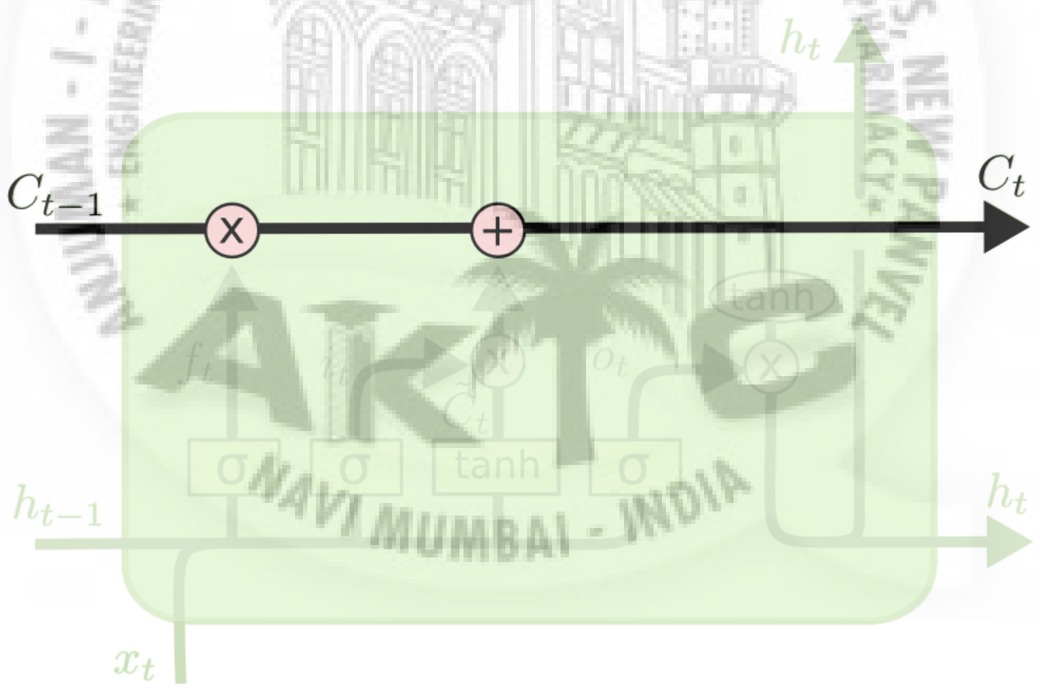
2.2.2 Disadvantages of Paper

a. Resource intensive training.

b. Suffer greatly from random weight initialization.

2.2.3 How to overcome the problems mentioned in Paper

a. Our model can give accurate results even if the data size is small.



2.3 Automatically Generating Novel and Epic Music Tracks.

The model (MuCyG) uses two discriminators and two generators. generators and discriminators based on the architecture used in MuseGAN, formed by CNN are used.

2.3.1 Advantages of Paper

a. Track-conditional Generation-Assumes that the bar sequence of one specific track is given and learn the temporal structure to generate the remaining tracks.

b. New musical notes can be generated from scratch.

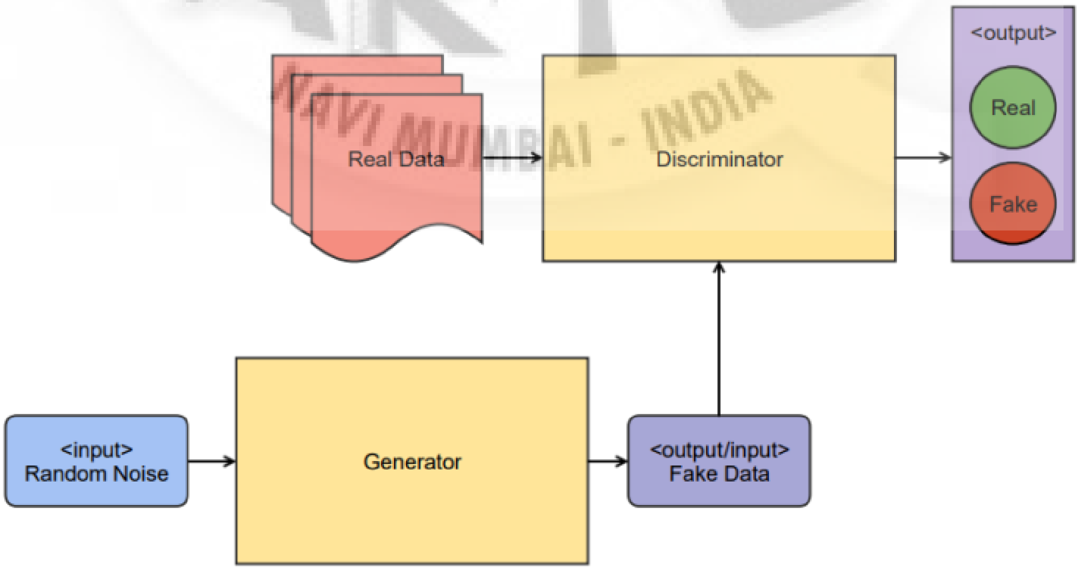
2.3.2 Disadvantages of Paper

a. Mode collapse: This case happens when the model over-specializes in only a small number of examples that can fool the discriminator.

b. Vanishing Gradient: A very common problem in other deep models resulting in a very slow training process or even stopping the process in a state far away from the equilibrium point.

2.3.3 How to overcome the problems mentioned in Paper

a. Training is not as intensive as GAN and there is no vanishing gradient



Chapter 3

Project Planning

3.1 Members and Capabilities

Table 3.1: Table of Capabilities

SR. No	Name of Member	Capabilities
1	Deepak Prajapati	Discriminator model
2	Ashraf Sonde	Generator model
3	Sufyan Shaha	Training and tuning

3.2 Roles and Responsibilities

Table 3.2: Table of Responsibilities

SR. No	Name of Member	Role	Responsibilities
1	Deepak Prajapati	Team Leader	Core development
2	Ashraf Sonde	Team member	Model development
3	Sufyan Shaha	Team member	Coding and Training

3.3 Assumptions and Constraints

Music will be generated at the end by artificial intelligence.

3.4 Project Management Approach

Agile methodology

3.5 Ground Rules for the Project

Make good quality code without bugs and greater accuracy in model.

3.6 Project Budget

INR 0

3.7 Project Timeline

6 months

1st month – research

2 – 3 month – developing discriminator and generator.

4th month – integration with MIDI library

5th month – solving bugs

6th month – training and tuning model

Chapter 4

Software Requirements Specification

4.1 Overall Description

Training of the GAN model is very GPU intensive.

4.1.1 Product Perspective

The Magenta team has used GAN and Transformers to generate music with improved long-term structure. In the Transformers' model, relative self-attention is used. It modulates attention which helps capture which self-referential phenomena exist in music.

4.1.2 Product Features

Attention takes two sentences, turns them into a matrix where the words of one sentence form the columns, and the words of another sentence form the rows, and then it makes matches, identifying relevant context. A neural network armed with an attention mechanism can actually understand what "it" is referring to. That is, it knows how to disregard the noise and focus on what's relevant, how to connect two related words that in themselves do not carry markers pointing to the other. Attention allows you to travel through wormholes of syntax to identify relationships with other words that are far away — all the while ignoring other words that just don't have much bearing on whatever word you're trying to make a prediction about.

4.1.3 User Classes and Characteristics

It can be used in assisting human musicians during the various steps of music creation: composition, arranging, orchestration, production, etc. Indeed, to compose or to improvise, a musician rarely creates new music from scratch. She/he reuses and adapts, consciously or unconsciously, features from various music that she/he already knows or has heard, while following some principles and guidelines, such as theories about harmony and scales. A computer-based musician assistant may act during different stages of the composition, to initiate, suggest, provoke and/or complement the inspiration of the human composer.

Music composers will focus on developing new music to give them ideas generated by the program whereas people who are more interested in listening to music will only be concerned with the music generated.

4.1.4 Operating Environment

Development environment:
Software requirements : python (used v3.7), tensorflow (v2.2 used), numpy, matplotlib, music21
Hardware requirements: 4 core CPU, RAM 16GB, GPU Tesla K80 12GB

Production environment:
Software requirements: python (v3.7 or above), music21
Since the output generated by the model is in the format '.midi' we will need music21 library to process and listen to the output.
Hardware requirements: 2 core CPU, RAM 4GB, audio output device.

4.1.5 Design and Implementation Constraints

While processing input we constantly would run out of memory. We had to process the input as small batches to avoid this problem. Since we used free tier cloud computing service (Google Colab) available to us, we would often run out of memory while training as the GPU that was randomly allocated to us was not sufficient to train a GAN.

4.2 System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

4.2.1 System Feature

1. Generator 2. Discriminator

Description and Priority

Both the features are of high priority.

Our generator is trained on musical data. After its training, it is used to generate new samples of data which are similar to the training data but completely new. The generator tries its best to mimic the original data.

Discriminator is trained to identify whether the data it receives is fake or not. After its training it is used to identify the inputs. these inputs are the outputs of the generator. the discriminator classifies them as real or fake and sends the feedback to the generator. Using this feedback the generator tries to produce even better outputs. The generator works to fool discriminator whereas the discriminator works to catch the generator.

Functional Requirements

Generator is used to generate new samples of data which are similar to the training data but completely new. Discriminator is used to identify the inputs. these inputs are the outputs of the generator.

4.3 External Interface Requirements

4.3.1 User Interfaces

Since the output is a '.midi' file it has to be run in python. We have chosen to generate the final output file by running command through terminal.

4.3.2 Hardware Interfaces

User will type the execution command to run the python script through keyboard to generate the output file that will be stored in the folder.

4.3.3 Software Interfaces

The program is compatible with every operating system. The tools used are Jupyter Notebooks, Google Colab. In addition the libraries used are Keras, TensorFlow, music21 and many others.

4.4 Nonfunctional Requirements

4.4.1 Performance Requirements

Two main components for the running the program are GPU and RAM. Processing a midi is memory intensive. We need to prepare batches for input, this will consume RAM. For faster performance, we run the model on a GPU as running on a CPU will be extremely slow.

4.4.2 Safety Requirements

The designed solution of the product is not harmful.

4.4.3 Security Requirements

Music21 is an open-source toolkit for Computer-aided musicology. It is licensed under the BSD license. Music21 is Copyright © 2006-2021, Michael Scott Cuthbert and cuthbertLab. Music21 code (excluding content encoded in the corpus) is free and open-source software. The music (if not the encodings) in the corpus are either out of copyright in the United States and/or are licensed for non-commercial use.

Chapter 5

System Design

5.1 System Requirements Definition

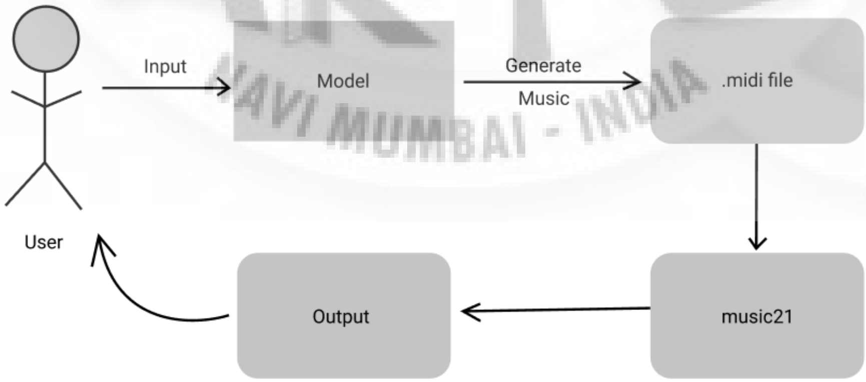
The program generates musical notes after learning from training set. Attention mechanism allows you to travel through wormholes of syntax to identify relationships with other words that are far away — all the while ignoring other words that just don't have much bearing on whatever word you're trying to make a prediction about

The objective of the requirements definition phase is to derive the two types of requirement:

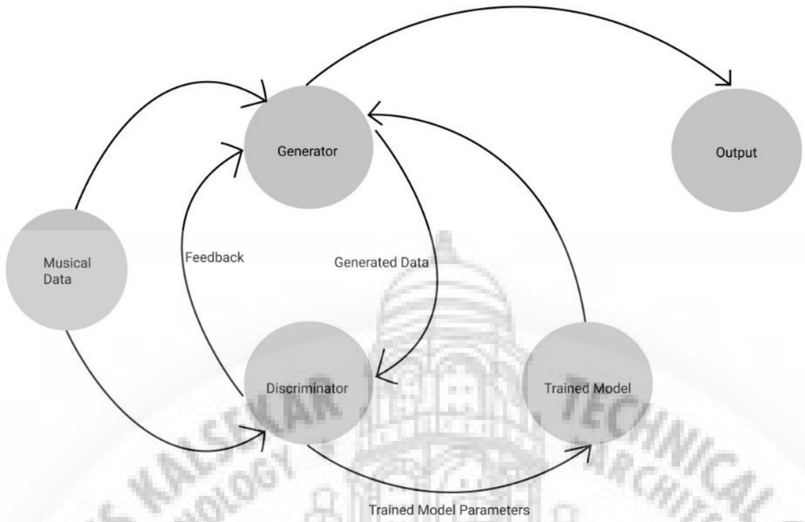
5.1.1 Functional requirements

The basic functionality is generate new music hassle free and without any human input.

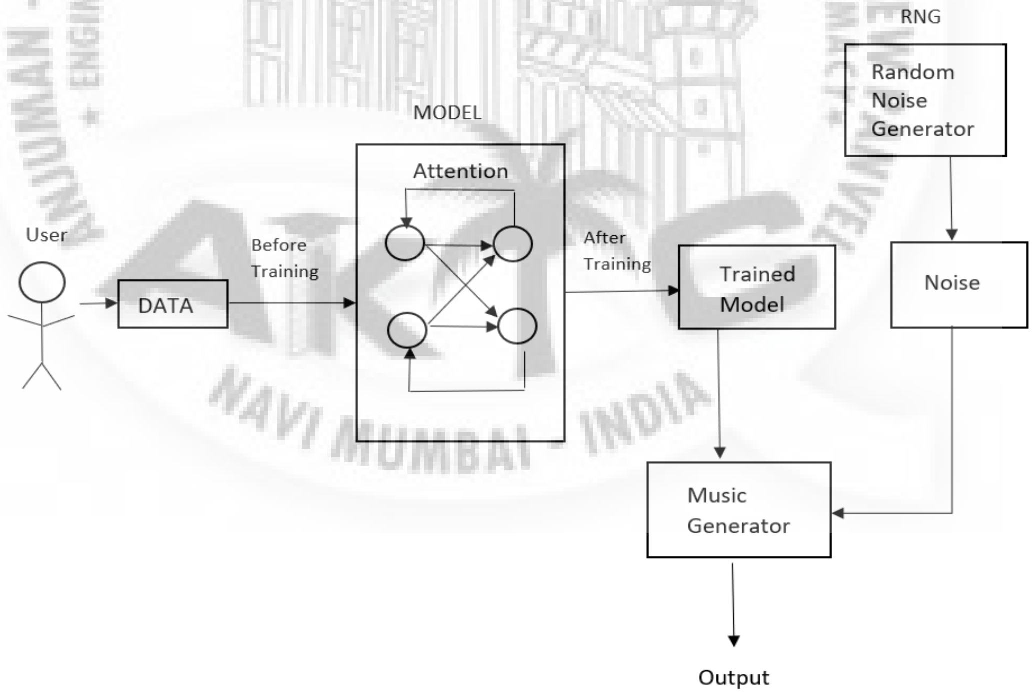
Use-case Diagram



Data-flow Diagram



5.2 System Architecture Design



5.3 Sub-system Development

We developed discriminator, generator and the attention mechanism.

5.3.1 Generator

Our generator is trained on musical data. After its training, it is used to generate new samples of data which are similar to the training data but completely new.

5.3.2 Discriminator

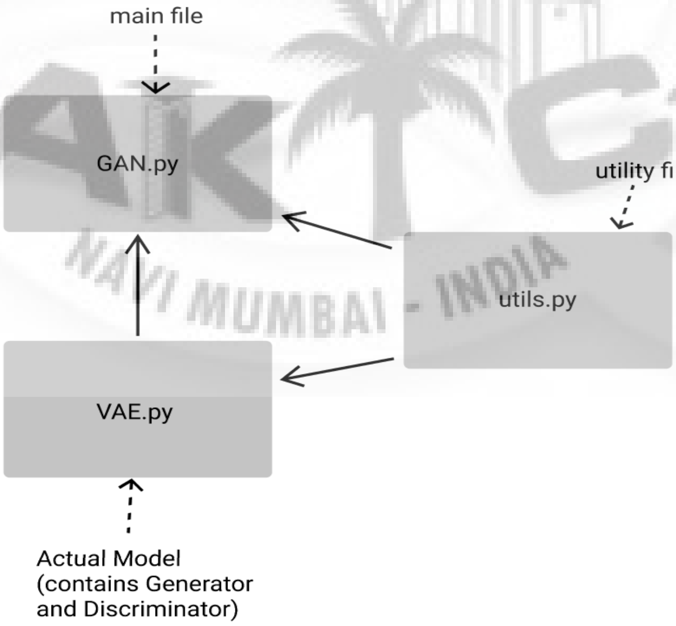
Discriminator is trained to identify whether the data it receives is fake or not. After its training it is used to identify the inputs. these inputs are the outputs of the generator.

5.3.3 Attention mechanism

Attention takes two sentences, turns them into a matrix where the words of one sentence form the columns, and the words of another sentence form the rows, and then it makes matches, identifying relevant context.

5.4 Systems integration

5.4.1 Component Diagram



Chapter 6

Implementation

6.1 Generator

```
gan.generator.summary()

Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
generator_input (InputLayer) [(None, 100)]             0
_________________________________________________________________
dense_1 (Dense)              (None, 3136)              316736
_________________________________________________________________
activation_4 (Activation)    (None, 3136)              0
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 14, 14, 64)        0
_________________________________________________________________
generator_layer_0 (Conv2D)   (None, 14, 14, 128)       204928
_________________________________________________________________
activation_5 (Activation)    (None, 14, 14, 128)       0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 28, 28, 128)       0
_________________________________________________________________
generator_layer_1 (Conv2D)   (None, 28, 28, 64)        204864
_________________________________________________________________
activation_6 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
generator_layer_2 (Conv2DTra (None, 28, 28, 64)        102464
_________________________________________________________________
activation_7 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
generator_layer_3 (Conv2DTra (None, 28, 28, 1)         1601
_________________________________________________________________
activation_8 (Activation)    (None, 28, 28, 1)         0
=================================================================
Total params: 830,593
Trainable params: 830,593
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.1: Generator

```python
def _build_generator(self):

    ### THE generator

    generator_input = Input(shape=(self.z_dim,), name='generator_input')

    x = generator_input

    x = Dense(np.prod(self.generator_initial_dense_layer_size), kernel_initializer = self.weight_init)(x)

    if self.generator_batch_norm_momentum:
        x = BatchNormalization(momentum = self.generator_batch_norm_momentum)(x)

    x = self.get_activation(self.generator_activation)(x)

    x = Reshape(self.generator_initial_dense_layer_size)(x)

    if self.generator_dropout_rate:
        x = Dropout(rate = self.generator_dropout_rate)(x)

    for i in range(self.n_layers_generator):

        if self.generator_upsample[i] == 2:
            x = UpSampling2D()(x)
            x = Conv2D(
                filters = self.generator_conv_filters[i]
                , kernel_size = self.generator_conv_kernel_size[i]
                , padding = 'same'
                , name = 'generator_conv_' + str(i)
                , kernel_initializer = self.weight_init
            )(x)
        else:

            x = Conv2DTranspose(
                filters = self.generator_conv_filters[i]
                , kernel_size = self.generator_conv_kernel_size[i]
                , padding = 'same'
                , strides = self.generator_conv_strides[i]
                , name = 'generator_conv_' + str(i)
                , kernel_initializer = self.weight_init
                )(x)
```

6.2 Discriminator

```
gan.discriminator.summary()

Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
discriminator_input (InputLa [(None, 28, 28, 1)]       0
_________________________________________________________________
discriminator_layer_0 (Conv2 (None, 14, 14, 64)        1664
_________________________________________________________________
activation (Activation)      (None, 14, 14, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0
_________________________________________________________________
discriminator_layer_1 (Conv2 (None, 7, 7, 64)          102464
_________________________________________________________________
activation_1 (Activation)    (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
discriminator_layer_2 (Conv2 (None, 4, 4, 128)         204928
_________________________________________________________________
activation_2 (Activation)    (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
discriminator_layer_3 (Conv2 (None, 4, 4, 128)         409728
_________________________________________________________________
activation_3 (Activation)    (None, 4, 4, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 2049
=================================================================
Total params: 720,833
Trainable params: 720,833
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.2: Discriminator

Code :

```
def _build_discriminator(self):

    ### THE discriminator
    discriminator_input = Input(shape=self.input_dim, name='discriminator_input')

    x = discriminator_input

    for i in range(self.n_layers_discriminator):

        x = Conv2D(
            filters = self.discriminator_conv_filters[i]
            , kernel_size = self.discriminator_conv_kernel_size[i]
            , strides = self.discriminator_conv_strides[i]
            , padding = 'same'
            , name = 'discriminator_conv_' + str(i)
            , kernel_initializer = self.weight_init
            )(x)
```

Chapter 7

System Testing

7.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Dimension of layers	Incompatible dimension	Training blocked	Training successful
T02	Input data dimension	Incompatible dimension	Model failed to take input	Model failed at training stage

7.2 Sample of a Test Case

Title: Dimension of layers

Description: Dimension of layers of generator did not match with that of discriminator.

Precondition: Training of the model stopped.
Assumption: Training should be successful.

Test Steps:

1. Browsed the stackoverflow for the problem
2. Found a similar solution
3. Implemented the solution using the reference.
4. Successfully trained the model

Expected Result: Training of the model to be successful.
Actual Result: Training successful.

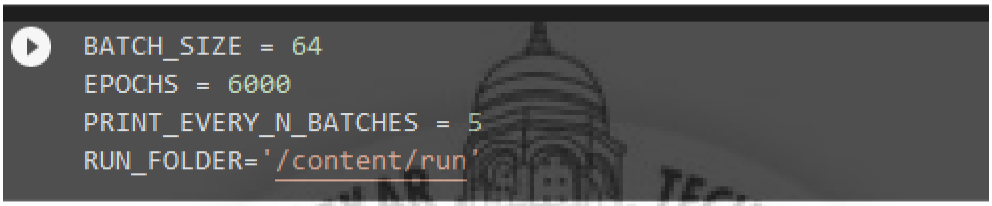
7.2.1 Software Quality Attributes

- Object oriented approach
- Code splitting
- Reusable functions
- Easy maintenance
- Can customize the size of generator
- Ease of use

Chapter 8

Screenshots of Project

8.1 Images of the project



Chapter 9

Conclusion and Future Scope

9.1 Conclusion

- We have presented a generative model for music generation using Neural Network.
- Melody generation from lyrics in music and AI is still unexplored well. Making use of deep learning techniques for melody generation is a very interesting research area, with the aim of understanding music creative activities of human.
- We will make a model that will generate music notes based on the previous and future notes using Attention mechanism.

9.2 Future Plans

- We plan to continuously improve performance and accuracy of our model.
- We will be experimenting with different loss functions and optimizers.
- We will be introducing more complex models and data representations that effectively capture the underlying melodic structure.

References

- MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment.
- Deep Learning Techniques for Music Generation – A Survey.
- Detecting Generic Music Features with Single Layer Feedforward Network using Unsupervised Hebbian Computation.
- Conditional LSTM-GAN for Melody Generation from Lyrics.
- GANSYNTH: ADVERSARIAL NEURAL AUDIO SYNTHESIS.

Achievements

NA