International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 08 | Aug 2022 www.irjet.net p-ISSN: 2395-0072

Application of Recurrent Neural Networks paired with LSTM - Music Generation

Rishabh Kumar [1], Ankit Mishra[2], Yuvraj Chhabra[3]

[1,2,3] *School of Computer Science and Engineering (SCOPE),VIT, Vellore.*

***Abstract*- Music is an integral part of a promotional video; it builds up the identity of a brand or entity. Some brands are also associated with the music that they've played in their catchy advertisements. Brands go to the extent of collaborating with Industry-leading music companies & artists in order to produce the beautiful music, which would go along with their advertisements.**

Composition and Production of Music, however, is relatively expensive. The sheer cost of hiring a talented team, capable of producing the iconic music for your company, might cost you a fortune. While, the larger companies, with a lot of financial resources may be able to afford such deals, while the other business owners, like Start-ups & small businesses suffer. This results in further polarisation of the already polar economy. The richer companies get richer, by attracting a larger range of audiences, while the smaller businesses suffer from a lack of customer reach. Looking at the situation, with a fresh set of minds, can help develop solutions, which were previously not thought about. Artificial Intelligence in the last decade has achieved what was beyond imagination for the Human Kind.

AI Music Generation platform can become a reliable and cost-effective solution, for a business owner to produce their own music. Artificial Intelligence can be a life-saver, not only in terms of the number of financial resources a company would have to potentially spend on producing their unique music, but also in terms of the amount of time spent and the efforts made from the company's side.

A web-based platform, which can be accessed across all the globe, would help the product gain customers without geographic limitations. AI algorithms can be trained to understand which combination of sounds make a pleasant-sounding tune (or music). This may be achieved using multiple Machine Learning Algorithms.

***Keywords*--- Neural Networks, Long Short-Term Memory networks, Artificial Intelligence, Music Theory, interactive music generation, web technologies, open-source software.**

I. INTRODUCTION

Music Theory is what musicians and academicians use to describe the phenomenon that they heard in a musical composition. Core concepts of music theory would help us comprehend further which set of sounds go together in order to produce melodious songs. The most fundamental terms one encounters while understanding the Music Theory are Melody, Harmony and Rhythm. However, the focus here would remain on the rudiments of music theory. These Rudiments of Music Theory are- Scales, Chords, Keys (or Note) & Notation. Our primary focus would be on Notes (or Keys) and Chords. Let's take the example of a Piano, a piano has something we call octaves, every octave has a set of 7 Musical Keys - C, D, E, F, G, A & B. Between every note, there is another Note, these notes are- C#, D#, E#, F#, G#, A# & B#. Every single note has a distinct sound. When multiple notes are played together in harmony, it is called a Chord. Chords can alternatively be defined as individual units of harmony.

A combination of chords and/or notes, which has a certain degree of Melody, Harmony and Rhythm is known as music. We aim to extract sequences of chords and notes, which sound harmonious.

[1] Neural Network can help achieve a Machine Learning model, capable of the task of comprehending the music and producing music. Neural Networks can be understood with a human example, when shown a drawing of a cat, even though it might be just a doodle, we can almost instantly recognize it as a cat. This is because we focussed on the "features" of the cat. The features of the cat include it having a couple of eyes, four legs, a couple of ears with a typical shape and so on. We cannot always be certain that an object is what the ML Algorithm predicted; thus, we assign a probability to the same. Thus, every feature of the cat is given a probability, which is later on converted into percentage. These features however aren't all equally important. After we find the relative importance of these features in terms of percentage, we call

it weight. Now, we multiply the percentage of features with the probability of each of the features to get a value. We apply the sigmoid function on this obtained value to get the value of probability of the object being a cat.

This single unit or function can be called as a neuron, it basically takes the inputs and multiplies them, then sums them up and applies the activation function to the sum. Neuron's goal is to adjust the weight based on a lot of examples of inputs and outputs. Neuron learns about weights based on inputs and desired outputs. Neural Network is a network of such neurons, neurons might have their input as output of another neuron or vice versa, this makes a neural network.

As human beings we don't think the same thing again and again. Rather we use our previous knowledge and try to understand it. Traditional neural networks can't do this; they can't refer to the previous information. So, to address this issue recurrent neural networks come into play where the information forms a network of loops so the same information is accessible again. There are times when we need more information to understand things like predicting the last word of a sentence, while we're typing a sentence, naturally it's not sure what it would be without more context. So that is when the difference between the actual word required and the point where it is required become more.[2] Long Short-Term Memory Network is a different type of RNN capable of learning those long gaps. LSTM is a chain like structure but with a different structure. It has four neural networks interacting in a special way rather than a single neural network.

Long Short-Term Memory Network can remember patterns of chords which sound harmonious. This in turn helps generate music, the Machine Learning Algorithm would be integrated with the Web Platform. The Website starts with helping the users understand the purpose of the application and help users navigate to a page, where selecting their preferable instrument and certain associated configurations, the configurations are fed into the Machine Learning Model, which then churns out the music using those parameters. The Machine Learning Model would already have been trained and deployed on a cloud platform. Music is generated and is returned to the user, who can then listen to it and decide whether they want to purchase the same.

II. LITERATURE SURVEY

Generating music requires an in-depth understanding of the composition of music. Notes & Chords can be called the building blocks of music [15]. It is well known since v. Helmholtz's (1877) ground breaking studies that the human auditory system cannot distinguish between acoustic stimuli with pitch frequencies f0, f1when their frequency ratios are separated by one octave [16]. Thus, every note can have different sound depending on the letter assigned to it (C, D, E, F, G, A & B) and the octave in which the note lies. There are certain patterns of music, which sound pleasant to ears, these can be called melodious patterns. Melodious music positively affects the mood of people around [14]. Melody is a linear succession of musical notes along time [30]. The universal symmetry in tonal music is transposition invariance (K¨ohler1969): A melody does not significantly alter its character, when played in a different key [16]. There are a number of methods, which produce music without any melody & a loop of notes [8].

From the perspective of music composition, piano music can be regarded as a sequence composed of multiple notes; this sequence is formed according to the relevant music theory rules, and each component note is dependent. Music language models are useful for a variety of music signal and symbolic music processing tasks, including music development, symbolic music classification, and automated music transcription (AMT) [4]. Sliding-Window is a necessary concept, when dealing with a large sequence [6]. Concepts of Machine learning can aid in building longer sequences of melodious patterns, to ultimately produce music [19].

Neural networks have been applied to raw data or low-level representations to jointly learn the features and classifiers for a task [5,9]. Simple neural networks may provide further insights into the computational properties of a highly trained and dynamic auditory system [13]. Deep Neural Networks have also been used for algorithms involving music [7,22]. A recurrent neural network is used to model the correlations between pitch combinations over time, this is basically the language model [5,29]. There is a long history of generating melody with RNNs. A recurrent auto predictive connectionist network called CONCERT is used to compose music (Mozer 1994) [30,26]. Multi-Layer Perceptron (MLP) neural network, complex-valued neural network (CVNN) & Support Vector Machines (SVM) neural networks are additional algorithms, which may be used to produce music [10,12].

Recurrent Neural Network (RNN) is one of the neural network models. Compared with other common feedforward neural networks, there are more feedback links between hidden layers in RNN [27]. These RNN's can help us build sequences of notes & chords, which sound melodious. Although the RNN model shows excellent performance in time series problems, the neural network model also has problems such as gradient explosion or gradient disappearance [23,24]. On this basis,

to avoid the limitations of the RNN in terms of gradients, the Long-Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are introduced [3,20,21].

Theoretically, any RNN can learn the temporal structure of any length in the input sequence, but in reality, as the sequence gets longer it is very hard to learn long-term structure [17,18]. Different RNNs have different learning capability, e.g., LSTM performs much better than the simple Elman network [30,11]. Thus, LSTM can be the ideal algorithm to recognize longer patterns to generate music [28]. Through the introduction of the LSTM network, some invalid information can be removed; at the same time, the preservation of the valid information is realized so that the performance of the RNN can be improved further [3,25].

It is thus concluded, that LSTM is one of the best algorithms to be used, to remove invalid information. LSTM also has the ability to recognize longer patterns. In the domain of recurrent neural networks, there are also a number of other algorithms, such as GRU, HRNN, etc. which can be used to achieve similar result. However, the crux of the findings is that LSTM is one of the best algorithms to implement, while producing music. The next section deals with understanding the problems that the application presented in this paper could potentially solve.

III. PROBLEM STATEMENT

Music is an essential need of various industries for a variety of needs like advertisements, social media handles, digital marketing etc. However, production of music is rather expensive and takes some time. Artificial Intelligence generated music could help small-businesses, start-ups, etc who want quality music within a certain time frame. This project aims to build a platform which can help people across the globe to generate their own music, using instruments of their choice and later purchase it. The person/company has sole rights over the music they produce, thus, they can build their own brand reputation with it.

Analysing the existing work on AI Music generation, certain things could be deduced from them. All the papers dealing with AI Music generation, firstly deal with Music Theory. A deep understanding of Music theory is necessary to work on the platform. Another point to be noted is that machine learning based approaches are required to generate music. Beyond this point, all the papers vary in their approaches. A majority of papers use the concepts of Convolution Neural Networks (CNN) in their implementation. However, a paper clearly mentions that a combination of Deep Learning Algorithms and *Long Short-Term Memory* networks improves the overall accuracy of the model, [22] In addition, the results show that the LSTM + DNN classifier improves music emotion recognition accuracy by 1.61, 1.61, and 3.23 points.

This helps us understand that Neural Networks are an appropriate fit for Models involving Music Theory. Some Papers also use different methods like complex-valued neural network (CVNN) or, K-Means Clustering or, Multi-Layer Perceptron (MLP) neural network and Support Vector Machines (SVM) neural networks as a sub-task of the entire Model. However, research indicates that the most efficient model for our AI Music generator is a combination of Deep Learning Algorithms and Long Short-Term Memory, which is based on recurrent neural networks (RNN).

There may be certain challenges involving the usage of DNN+LSTM based on RNN, the first one being One-to-Many Sequence Problem where input data has one time-step and the output contains a vector of multiple values or multiple time-steps. One of the solutions have been discussed in the research paper that has been analysed. It stated that using Stacked LSTM, the One-to-Many Sequence problem may be avoided. Many-to-Many sequence problem may also be faced, which may be avoided in similar ways as One-to-Many Sequence Problem.

The Overall Aim is building a platform, which can be accessed across the globe. The platform shall let the users pick their own customizations, the instrument, and certain other factors from Music Theory, which then will be used to run the Machine Learning Model built on Deep Learning Algorithms and Long Short-Term Memory. The Model would be trained prior to this, using sounds of various instruments. The ML model would produce music after learning the various combinations of chords which sound melodious. The users may then purchase the music, which would then belong to the user or the company and may be used by them in various commercials, their social media handles and Digital Marketing.

IV. RESEARCH FRAMEWORK

The existing research has successfully given a deeper understanding about how to develop an AI Music Generation platform. It has also given better knowledge & crisp understanding about the various machine learning available for the task. The platform is built on the web, which would make it easily accessible to the people across the globe. The platform would guide users through the simple and intuitive user interface, which would help users select their choice of musical

instrument, then proceed to fine tune certain parameters, which would be chosen by the user. The parameters selected by the user and the instrument are then fed into the machine learning algorithm. The algorithm then processes the data, to produce a song. The song is returned back to the client, and the client then has the option to firstly listen to the song and then, buy the same, if it hasn't already been bought.

The Machine Learning component of the platform needs a significant amount of research, due to variety of approaches available for the same. After analyzing a variety of ML Models, which are available for generation of music and looking at the pros and cons of a variety of methods, it is decided that the concept of Neural Networks would be used. Recurrent Neural Networks would be required, these Neural Networks basically perform the same function at every step, the input of every single neuron however is dependent on the output of the previous neuron. Long Short-Term Memory Network (LSTM), which is a type of Recurrent Neural Networks can help in remembering patterns or sequences of chords, which sound melodious one after the other. This section helps in understanding how the components work independently & in combination using a detailed architecture diagram.

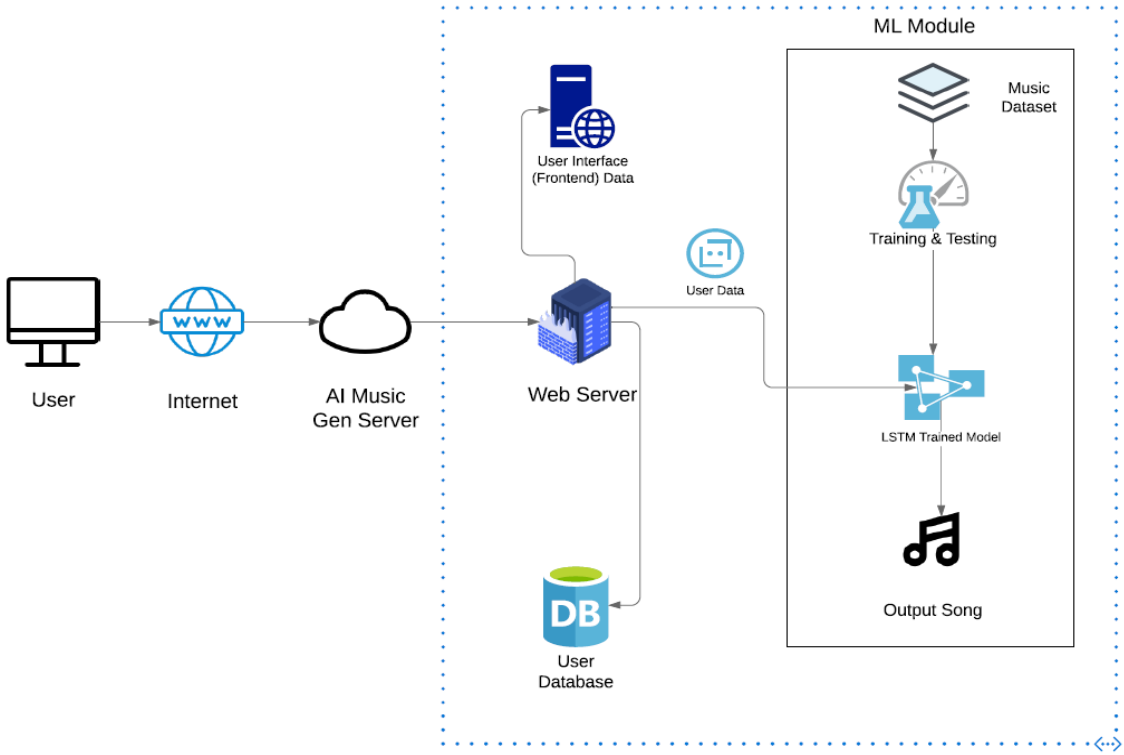

Fig. 1 Architecture Diagram of AI Music Generator

V. METHODOLOGY

The architecture diagram shows how the various modules function together, an in-depth understanding of the same is presented in this section. The overall description of the function is done firstly, after which every module is covered in great depth. The users, who is connected to the internet sends a request to the server module of our application, the server module sends back the website data to the user, after it fetches the same from the frontend module. The user can now send their credentials & fine tune their instrument along with a certain parameter to generate data. The user again sends back data to the server module, which firstly checks the user's identity then goes ahead to send the parameters to the ML Module. The ML Module runs the LSTM Algorithm which has been trained on data of various musical instruments. The ML Algorithm returns the newly generated music back to the user & the user is given an option to purchase the same. The entire system can be divided into the following modules.

A. ML Module

The ML Module, which is run on a cloud-based server, is the component that produces music. It produces the music using the Long Short-Term Memory Network (LSTM) Algorithm, which is based on neural networks. Neural Networks are built with the basic building block of neurons. Neural Networks can be understood with a human example, when shown a drawing of a dog, even though it might be just a doodle, we can almost instantly recognize it as a dog. This is because we focused on the features of the dog. The features of the dog include it having a couple of eyes, four legs, a couple of ears with a typical shape and so on. We cannot always be certain that an object is what the ML Algorithm predicted; thus, we assign a probability to the same. Thus, every feature of the dog is given a probability, which is later on converted into percentage. These features however aren't all equally important. After we find the relative importance of these features in terms of percentage, we call it weight. Now, we multiply the percentage of features with the probability of each of the features to get a value. We apply the sigmoid function on this obtained value to get the value of probability of the object being a dog. This single unit or function can be called as a neuron, it basically takes the inputs and multiplies them, then sums them up and applies the activation function to the sum. Neuron's goal is to adjust the weight based on a lot of examples of inputs and outputs.

Neuron learns about weights based on inputs and desired outputs. Neural Network is a network of such neurons, neurons might have their input as output of another neuron or vice versa, this makes a neural network. As human beings we don't think the same thing again and again. Rather we use our previous knowledge and try to understand it. Traditional neural networks can't do this; they can't refer to the previous information. So, to address this issue recurrent neural networks come into play where the information forms a network of loops so the same information is accessible again. There are times when we need more information to understand things like predicting the last word of a sentence, while we're typing a sentence, naturally it's not sure what it would be without more context. So that is when the difference between the actual word required and the point where it is required become more.

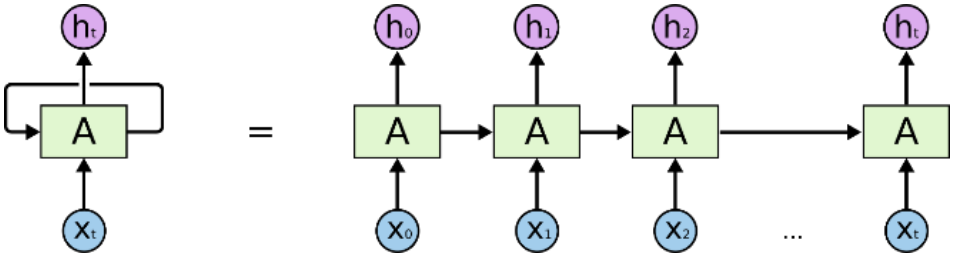

Fig. 2 LSTM Network

LSTM is a chain like structure but with a different structure than traditional neural networks. It has four neural networks interacting in a special way rather than a single neural network. Long Short-Term Memory Network can remember patterns of chords which sound harmonious. The ML Module is firstly trained using existing instrumental music. After the Module has been trained, we input certain parameters into the ML Module, the parameters are there to ensure that every time the ML Model is run, the results do not come out to be the same. The ML Module after being run produces certain music, which are sent back to the server, which then sends the data back to the User.

B. Server Module

Server Module helps coordinate all the activities happening in the entire system, it keeps a track of the user requests, on receiving the requests, it replies back with the appropriate information. The Server module here is basically a cloud-based platform, upon which all the services like the database, the entire user interface (frontend) & the ML Module are running. The server is to be built on a cloud-based platform. Some of the most popular cloud-based platforms are Google Cloud Platform & Amazon Web Services. JavaScript based server module would be built to help coordinate all the tasks.

The user initially requests for data from the webserver, which then returns the user interface of the website back to the user. The user further interacts with the user interface & enters his/her credentials. The credentials are then sent back to the server module. The server module checks the credentials from the database module & returns a response accordingly. Once the user has logged in, he/she gets the option to generate music with certain parameters, the parameters are again sent to the Server Module, which then invokes the ML Module to produce music with these specific parameters. The ML Module produces music, which is then sent to the server, the server stores the music file temporarily & also sends the

music back to the user, the user may listen to the music using their user interface, if the user wants to proceed, the user may purchase the music. If the user requests the server to purchase the music with digital rights, the user is redirected to the payment's portal, the server fetches the payment portal from the frontend module & sends it back to the user's frontend. The user may enter his payment details & purchase the digital rights from that point on.

C. Frontend Module

The module helps the user interact with the server, which further coordinates the various functionalities of the entire system. The frontend would be built on the latest technologies in the field of web development. HTML5, CSS3 & JavaScript would be used to build the entire user interface, which would help give the user-interface a modern look. The website would also be made responsive using the popular Bootstrap framework.

The user receives the Frontend module, when he/she sends the request to server. The landing page would help the user understand about the entire system & the functionalities of the website. The user can navigate to the page where he/she requests for an instrument & defines certain parameter. This parameter might be any random number, which is fed into the ML Model through the server module to produce a unique music every single time. Once the Music is produced, the user is asked to either login or register. The credentials of the user are then sent over to the server. The server cross-verifies the credentials with the backend module & then sends back a response accordingly. The user can then redirect himself/herself to the payments page & purchase the digital rights to the newly generated music.

D. Database Module

The Database module stores the credentials of the users. The database also holds an account of the digital rights of the music with the parameters associated with it like the instrument & it's generation parameters. The records help in ensuring that the same music cannot be sold twice to another organization or individual.

When the user enters the login page, the credentials are sent over through the user interface to the server using REST Protocols, the sent data is cross verified by the server using the records of the user database. Once the user is authenticated, the success message is sent, otherwise an error code is generated & the user is told to retry the authentication process. Thus, the Database module helps in maintaining a record of all the consumers & works on the REST based protocol service. The following illustration helps us in understanding how the data flows throughout the application & how the modules work together to perform the various functionalities of the system.

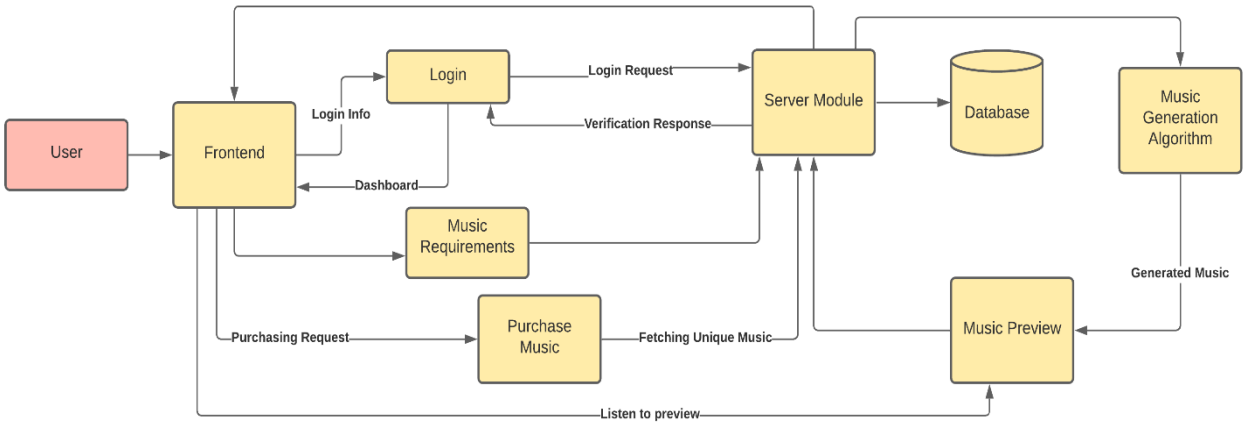

Fig. 3 Data Flow Diagram of AI Music Generator

The user interacts with the website through the frontend from the browser which is accessed from the database where the website is stored on the cloud on some hosting platform. Now, Through the website, the user will have to log in with his/her username and password which are in the database on the cloud, now this Login information will be verified from it and the user will be allowed to access the content which he already purchased, his previous generated music preview and other information. Now for generating the music, the user needs to define his requirements like the type of instrument, type of melody, and certain other factors which will be sent to the server over the cloud and will be fed to the Music Generating Machine Learning Algorithm on the cloud which will generate the music according to the pre-trained model

present on the cloud. On generating the required music, Now the generated music will be stored in the database through the server which will enable the user to listen to it whenever he wishes and then can make a decision to buy it or not. This music will remain in the database till someone else or the initial user doesn't buy it with a unique copyright id.

Suppose, the user decides to purchase the music then the user can go to the payment option where and will be redirected to a payment portal through the server. Now as the payment gets completed the music with a unique copyright ID will be assigned to the user and the user will be allowed to download the final version of the music for this personal or commercial purposes.

VI. IMPLEMENTATION

The Implementation can be broken down into three major portions. The Three Breakdown of the implementation is:

1. Constructing the ML Model.
2. Training & Prediction
3. Deployment of the Model on Website

A. Constructing the ML Model.

The following pseudo-code can help building the LSTM Model, which would be used to generate the music:

Importing all the Dependencies

Music 21- A Toolkit for Computer-Aided Musical Analysis.

glob- In Python, the glob module is used to retrieve files/pathnames matching a specified pattern. The pattern rules of glob follow standard Unix path expansion rules.

pickle- "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

numpy - NumPy is a Python library that provides a simple yet powerful data structure

keras - It was developed to make implementing deep learning models as fast and easy as possible for research and development.

```
In [1]:
from music21 import converter, instrument, note, chord, stream
import glob
import pickle
import numpy as np
from keras.utils import np_utils
```

Pre-processing all Files

The file is first converted into a stream.Score Object inside every single music file, every element is checked, if it's a Note or a Chord. The Note or the Chord are stored in an array called as Notes. If the element is a chord, It is split into corresponding notes & concats them.

In [2]:
```
notes = []

for file in glob.glob("Classic_Guitar/*.mid"):
    midi = converter.parse(file) # Convert file into stream.Score Object

    print("parsing %s"%file)

    elements_to_parse = midi.flat.notes

    for ele in elements_to_parse:
        # If the element is a Note,  then store it's pitch
        if isinstance(ele, note.Note):
            notes.append(str(ele.pitch))

        # If the element is a Chord, split each note of chord and join them with +
        elif isinstance(ele, chord.Chord):
            notes.append("+".join(str(n) for n in ele.normalOrder))
```

Checking the Number of notes in all the files in the MIDI Music Files

In [3]:
```
len(notes)
```

We Open the Filepath to later on save the Notes in a File. This Process ensures that the MIDI Files do not need to be parsed again & again to find their notes, while improving the model further in the future.

In [4]:
```
with open("notes_classical_guitar", 'wb') as filepath:
    pickle.dump(notes, filepath)
```

Now the Notes are stored in the file "notes_classical_guitar"

In [5]:
```
with open("notes_classical_guitar", 'rb') as f:
    notes= pickle.load(f)
```

n_vocab stores the length of the unique notes

In [6]:
```
n_vocab = len(set(notes))
```

Prepare Sequential Data for LSTM

We have to build a Supervised Learning Model

Sequence Length- How many elements LSTM input should consider.

Basically, LSTM Model would take in 100 Inputs & predict one output

In [9]:
```
sequence_length = 100
```

PitchNames stores all the Unique Notes (& Chords) in a sorted order.

In [10]:
```
pitchnames = sorted(set(notes))
```

Mostly ML Models do work with strings.So, we convert the data into Integers.

The Following is mapping between Element (Note/Chord) to Number (Integer)

The Mapping is stored in a Dictionary

In [11]:
```
ele_to_int = dict( (ele, num) for num, ele in enumerate(pitchnames) )
```

We define Network_Input & Network_Output Arrays

In [12]:
```
network_input = []
network_output = []
```

In the following, seq_in is a set of 100 Elements (Notes/Chords) & Seq_oug is the 101th Element. This continues till all the Notes have been either a part of seq_in or seq_out

Now, The Notes array contains strings, however, we have to work with Numerical Data, so we convert the entire data into Numericals in the following code.

In [13]:
```
for i in range(len(notes) - sequence_length):
    seq_in = notes[i : i+sequence_length] # contains 100 values
    seq_out = notes[i + sequence_length]

    network_input.append([ele_to_int[ch] for ch in seq_in])
    network_output.append(ele_to_int[seq_out])
```

We are assigning n_patterns as the length of Network_Input

In [14]:
```
# No. of examples
n_patterns = len(network_input)
print(n_patterns)
```

LSTM Needs Input in 3 Dimension.

Now, we have Network_input & Network_Output

We need another Dimention, So we just take 1 as another dimention. This gives us the desired shape for LSTM.

In [15]:
```
# Desired shape for LSTM
network_input = np.reshape(network_input, (n_patterns, sequence_length, 1))
print(network_input.shape)
```

We Normalize the Data in the Next Step. Normalized Data is needed for Deep Learning Models, such as this one

In [16]:
```
normalised_network_input = network_input/float(n_vocab)
```

Network output are the classes, encode into one hot vector

Network Output are seperated, It is turned into one vector, using Keras np_utils.to_categorical

In [17]:
```
network_output = np_utils.to_categorical(network_output)
```

Create Model

Importing Necessary Libraries

In the previous section, Data has been made from our music, this section deals with feeding the converted data into the Music. The **Sequential model**, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away). Layers are the basic building blocks of neural networks in Keras. A layer consists of a tensor-in tensor-out computation function (the layer's call method) and some state, held in TensorFlow variables (the layer's weights). A **callback** is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc).

In [20]:
```
from keras.models import Sequential, load_model
from keras.layers import *
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

We have declared the model to be a sequential model. Since, it is the first later, we have to define the shape of the Input. We use the Shape of Normalized_network_input Sequence has to be returned, as this is not the last layer & further layers exist. **Dropout** is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. Dropout is easily implemented by randomly selecting nodes to be dropped-out with a given probability (e.g. 20%) each weight update cycle. This is how Dropout is implemented in Keras. Dropout is only used during the training of a model and is not used when evaluating the skill of the model. Application of dropout at each layer of the network has shown good results. Another LSTM Layer is added with 512 Units & Return Sequence is true, as this is another layer which is not the output layer. In the last Layer, Again the Units 512 & Return sequence is false by default. At the end, there's a dense layer of 256 Layers & a dropout of 0.3. In the final layer, The Dense layer should have units equal to n_vocab. it should have its activation function as softmax.

Softmax is typically used as the activation function when 2 or more class labels are present in the class membership in the classification of multi-class problems. It is also a general case of the sigmoid function when it represents the binary variable probability distribution.

In [21]:
```
model = Sequential()
model.add( LSTM(units=512,
              input_shape = (normalised_network_input.shape[1], normalised_network_
              return_sequences = True) )
model.add( Dropout(0.3) )
model.add( LSTM(512, return_sequences=True) )
model.add( Dropout(0.3) )
model.add( LSTM(512) )
model.add( Dense(256) )
model.add( Dropout(0.3) )
model.add( Dense(n_vocab, activation="softmax") )
```

Now, The Model is compiled, using the Adam Optimizer

The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.

The Model tries to minimize the Categorical crossentropy loss value.

Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

In [22]:
```
model.compile(loss="categorical_crossentropy", optimizer="adam")
```

A Summary of the Model can be displayed using the following code:

In [23]:
```
model.summary()
```

The following code block stores the Model in the file "model_classical_guitar.hdf5", while the loss is monitored. The Mode is Minimum, as the loss is to be minimized.

The statement below the same (model.fit) is used to start the actual training.

The Various parameters like epochs define the number of interations of the training & the other parameters which will be used to train the model.

In [24]:
```
checkpoint = ModelCheckpoint("model_classical_guitar.hdf5", monitor='loss', verbose=
```

```
model_his = model.fit(normalised_network_input, network_output, epochs=10, batch_siz
```

Now, we load the model to start making new music.

The Training has been done through Google Colab, to ensure that the Model Trains quickly.

In [25]:
```
model = load_model("model_classical_guitar.hdf5")
```

B. Training & Prediction

The following code is used to train the LSTM Model, which then can be used in the to Produce Music.

Predictions

Network_Input is now an np array, it is thus declared again

Network_Input here is basically a list of lists

In [26]:
```
sequence_length = 100
network_input = []

for i in range(len(notes) - sequence_length):
    seq_in = notes[i : i+sequence_length] # contains 100 values
    network_input.append([ele_to_int[ch] for ch in seq_in])
```

The Start is initialized with a random number, to generate different music every time. Integer to Element Mapping is created, as the Element to Integer Map was declared above. A dictionary is used again. Initial Pattern the starting index

from where, the 100 Notes (or Chords) are taken 200 Elements are generated, which should produce music for around 1 min. In the for loop, initially 100 elements are taken from the start point, then prediction is made using LSTM, Now, the 1st Element is removed & the 1-101 elements are used to carry out the same process. Prediction input is used to get the data into the standard format, suitable for the model. Prediction gives the probability of each unique element. The Element (Note/Chord) which has the Max Probability is taken.

The Result is then appended to the output & the pattern is changed to include the next 100 elements, not considering the first element. The recent value predicted by the Model is Appended next.

```
# generate 200 elements
for note_index in range(200):
    prediction_input = np.reshape(pattern, (1, len(pattern), 1)) # convert into nump
    prediction_input = prediction_input/float(n_vocab) # normalise

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

    idx = np.argmax(prediction)
    result = int_to_ele[idx]
    prediction_output.append(result)

    # Remove the first value, and append the recent value..
    # This way input is moving forward step-by-step with time..
    pattern.append(idx)
    pattern = pattern[1:]
```

Create Midi File

The following code section basically iterates over the Output Elements & Checks whether they are Notes or chords. *Output_notes* is used to store the output Notes or Chords. If the Element is a note, A note object is made, using the class note in the note sub package. If Note is found, it is given an Offset & Instrument is set from a list of Instruments in the Music 21 Library

Output is appended with this New Note along with its offset. Else, if it's Chord. If + is a part of the patter, or the patter is a complete digit, then the pattern is a chord. Every single Note is separated out by checking where the '+' sign is. Thus, getting all the notes of the chord. The array **temp_notes** is used to store the data of every single note & then further the list of notes (temp_notes) are converted into the corresponding chord object as new_chord. The Note is then appended into the output_notes array.

Creating a Stream Object from the generated Notes

Writing the Stream Object into a test_output file.

```
In [29]: offset = 0 # Time
         output_notes = []

         for pattern in prediction_output:

             # if the pattern is a chord
             if ('+' in pattern) or pattern.isdigit():
                 notes_in_chord = pattern.split('+')
                 temp_notes = []
                 for current_note in notes_in_chord:
                     new_note = note.Note(int(current_note))  # create Note object for each n
                     new_note.storedInstrument = instrument.Piano()
                     temp_notes.append(new_note)


                 new_chord = chord.Chord(temp_notes) # creates the chord() from the list of n
                 new_chord.offset = offset
                 output_notes.append(new_chord)

             else:
                     # if the pattern is a note
                 new_note = note.Note(pattern)
                 new_note.offset = offset
                 new_note.storedInstrument = instrument.Piano()
                 output_notes.append(new_note)

             offset += 0.5
```

In [30]:
```
# create a stream object from the generated notes
midi_stream = stream.Stream(output_notes)
midi_stream.write('midi', fp = "test_output.mid")
```

In [31]:
```
midi_stream.show('midi')
```

C. Deployment of the Model on Website

A Flask Based Web-app can be easily built using the above ML Code. This Web based application ensures better user interaction as well as a much larger potential audience, due to the ease of using a web-based application.

The deployment of the ML Model on a Flask Web-app can be boiled down to a few simple steps.

1) ***Compiling the ML Model into a simple Python Function:*** Defining a function within a new python file, which has only the Prediction portion of the ML Model previously written. The Model & the Notes extracted from the training set can be used through the concepts of file handling & need not be defined again in this python file.
2) ***Connecting the Webpages & the Model:*** Connecting the function using the app.py file & rendering the corresponding webpage. The code for the same is given below.

```
from flask import Flask, render_template, request
import Music_Generation
import Guitar_Music_Generation
app=Flask('Music_Generation')
@app.route('/')
def show_home():
   return render_template('index.html')
@app.route('/instrument')
def show_instrument():
   return render_template('Instrument.html')
@app.route('/piano')
def play_piano():
   #Music_Generation.generate()
   return render_template('piano.html')
@app.route('/guitar')
def play_guitar():
   Guitar_Music_Generation.generate()
   return render_template('guitar.html')
@app.route('/buy_piano')
def buy_piano():
   return render_template('payment_piano.html')
@app.route('/buy_guitar')
def buy_guitar():
   return render_template('payment_guitar.html')
app.run("localhost", "9999", debug=True)
```

Implementation Screenshots

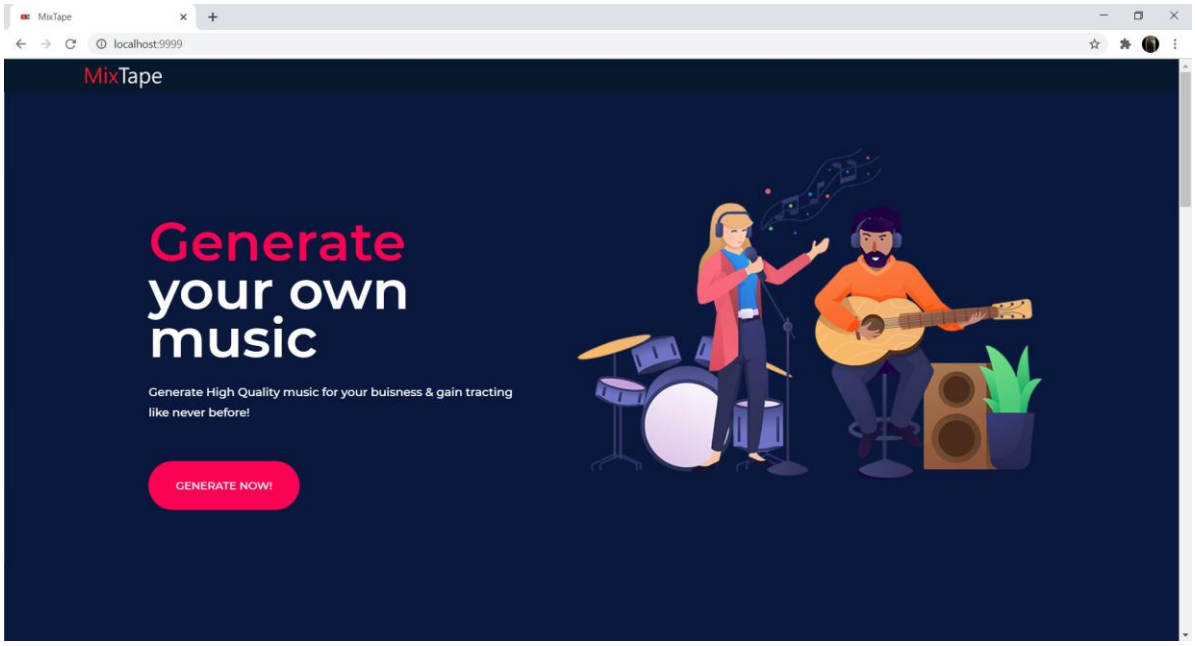

Fig. 4 Landing Page

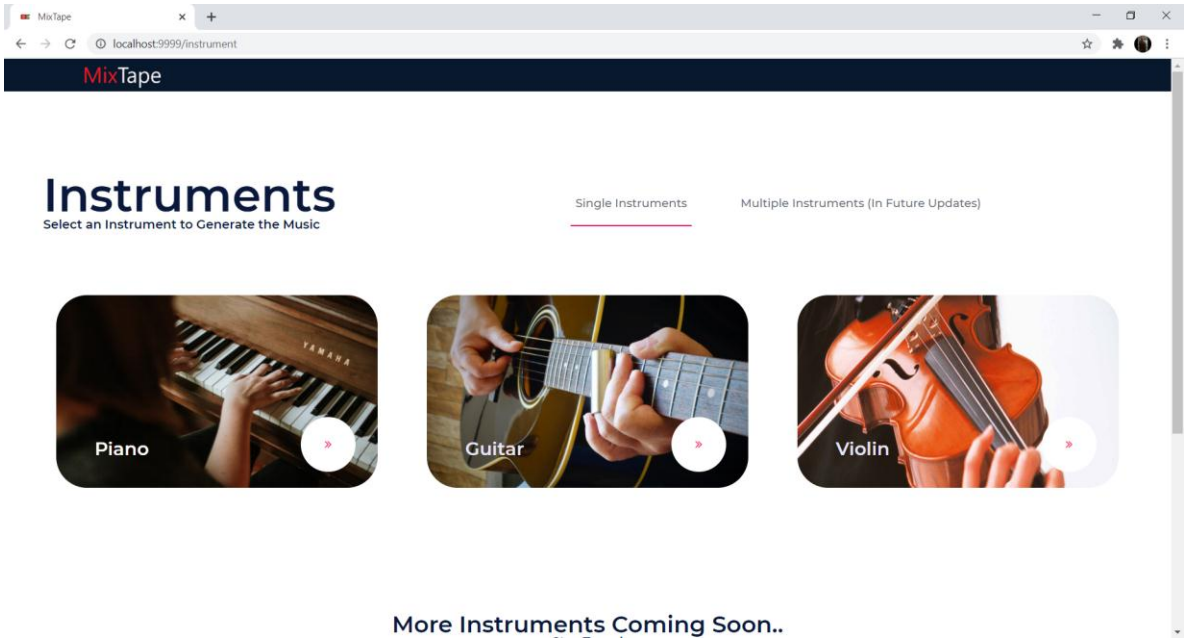

Fig. 5 Instruments Page - Selection of the Instrument of User's Choice

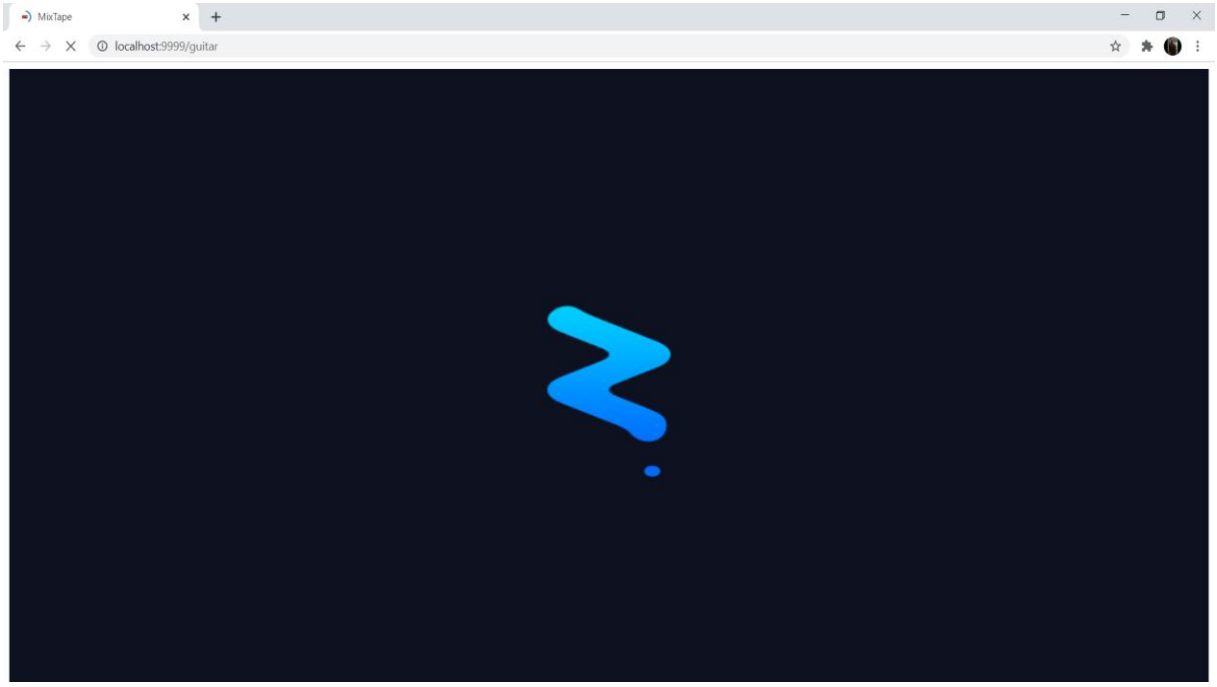

Fig. 6 AI Generating Music as the Loading Animation is displayed

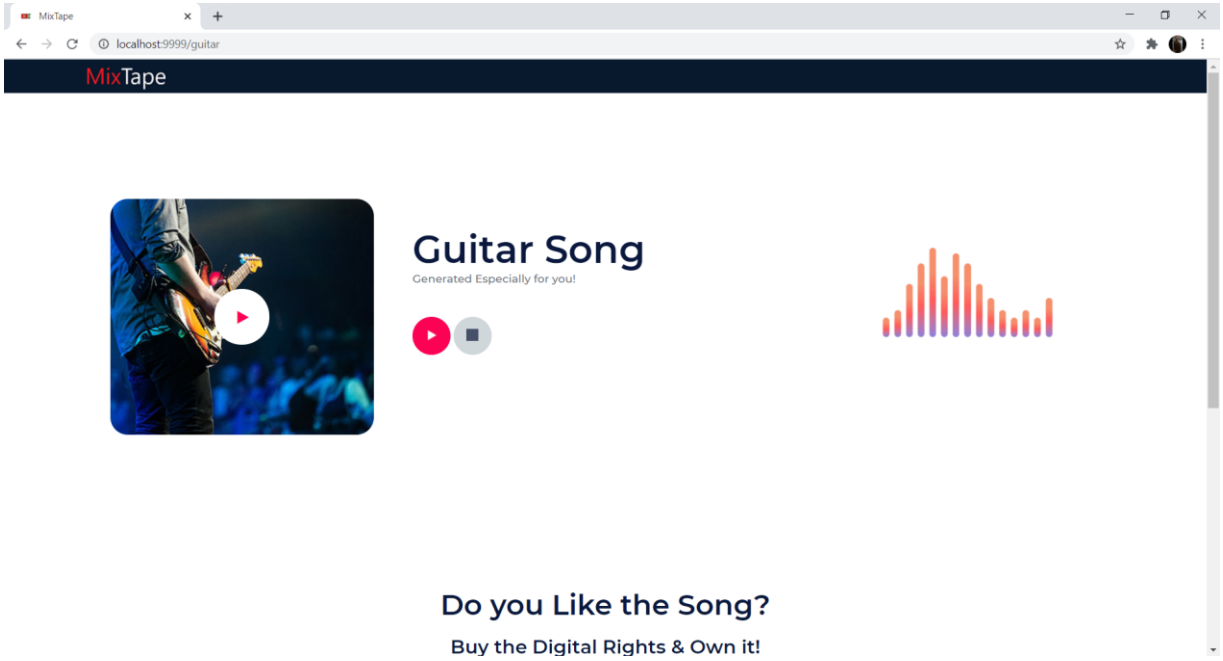

Fig. 7 AI Music Page- Users may listen to the generated Music & Purchase the Digital Rights

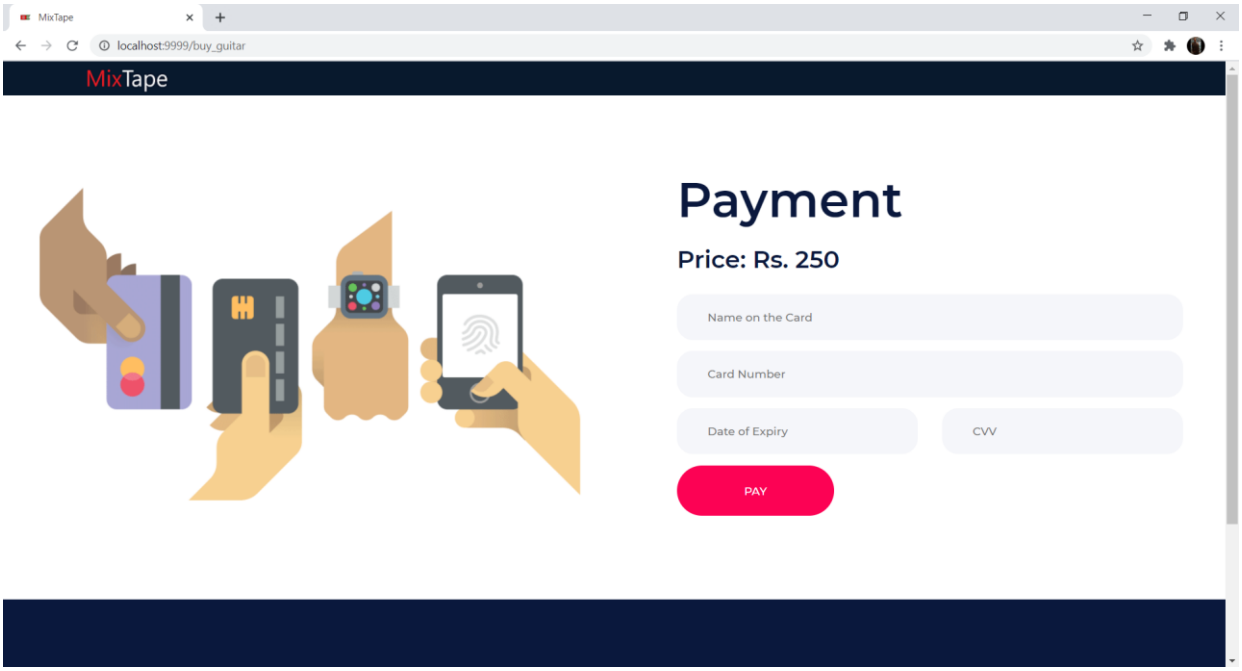

Fig. 8 Payment Page for users to purchase the AI Generated Music

VII. TESTING

The AI Music Generator generates music by recognizing the pattern of Notes & Chords of the Dataset that is used as the training set. The dataset does not have any output variable or "prediction" as such. The testing of the model is done by varying the various parameters, which are used in the training, testing & prediction phase of the Machine Learning. The parameters for the deployment (or the flask app) do not have any effect pronounced, that are worth noting. This section deals with the various inter-related parameters of the Machine Learning Model & finding a relationship between them. Graphical analysis using suitable kind of plots has further been done, to establish a relationship between the two parameters being analysed.

A. Sequence Length & No. of Unique Notes (or, Chords) being produced on average by the Machine Learning Algorithm.

TABLE I
SEQUENCE LENGTH V/S NO. OF UNIQUE NOTES

Sequence Length	Unique Notes/ Chords (Average Value)
100	35
200	43
300	27
400	26
500	32

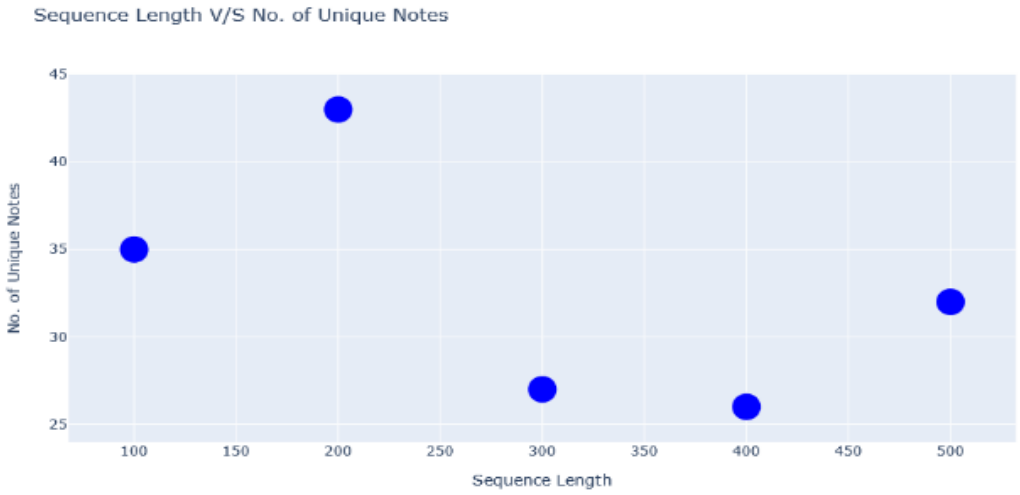

Fig. 9 Sequence Length V/S No. of Unique Notes

To interpret this result, the meaning of Sequence Length shall be understood. Sequence length is the length of Elements, that would be fed into the LSTM Algorithm in one iteration, looking at the pattern of a certain number of elements, the Algorithm produces a Music Note/Chord. One can imagine this like a window of 'n' elements. The Window keeps moving, removing one note from the rear end & taking in the new note it itself produced.

The greater the value of sequence length, the better the result should be. However, Analysis shows that the Most no. of ***unique notes are produced, when the sequence length is 200***. The greater is the number of unique Notes (or chords) in a music of fixed length, the better it sounds reveals primary analysis for this particular algorithm. **This means, given that a system is having sufficient Computing resources, 200 shall be the Idea Sequence length to run the Algorithm**. Analysis also shows that the Time (t) to run the algorithm is directly proportional to the Sequence Length, the same is shown in the next sub-section.

The conclusion from all the information above is, that the algorithm shall Ideally be using the Sequence Length of 200, however, ***Sequence Length of 100 produces music, which is roughly of the same quality & would give users half the waiting time****. Thus, the AI Music Generator uses a sequence length of 100.*

B. Time to Execute the Algorithm varying with the Sequence Length

As mentioned in the previous sub-section, Time to execute the algorithm increases with the Sequence Length. This section shall justify the reason for taking up the sequence length of the Algorithm as 100, in numeric terms.

TABLE II

EXECUTION TIME (IN SECONDS) V/S SEQUENCE LENGTH

Sequence Length	Execution Time (in seconds)
100	37.65
200	66.49
300	93.32
400	121.14
500	160.5

The Execution time increases by almost 60 seconds, when the sequence length increases by 100. Sequence length of 100 produces sufficient number of unique notes (or chords), which is enough to make the music harmonious.

Sequence length is the number of elements (notes or chords) from the training dataset (or input music), which are taken inside one iteration of the Algorithm. This can be imagined as an algorithm where 'n' number of elements are taken inside a window to produce one element.

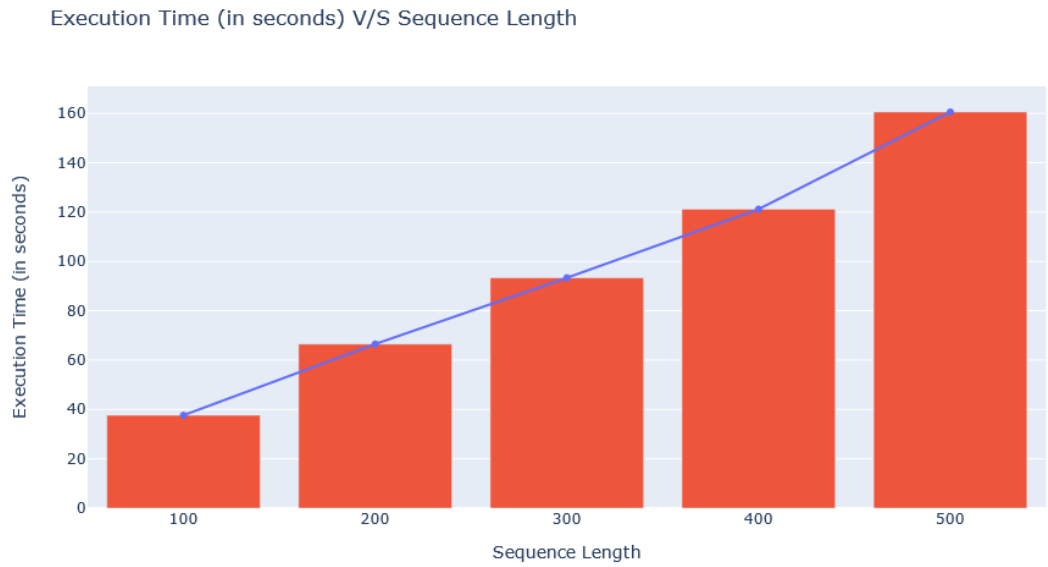

Fig. 10 Execution Time (in seconds) V/S Sequence Length

A survey by Brafton shows the average duration of session of a user on a website is close to 2 Minutes and 17 seconds, thus, optimizing both the Execution Time & the quality of the music produced is a must. After taking all the parameters into consideration, it is concluded that the AI Music Generator would be generating music using the sequence length of 100.

C. Varying the No of elements to be produced (in a single iteration) to change the duration of music produced.

Increasing the No. of Elements to be produced in the Prediction Algorithm leads to increase in duration of the music generated. The same has been noted in the following table.

TABLE III

DURATION (IN SECONDS) V/S NO. OF ELEMENTS

No of Elements	Duration (Seconds)
100	26
200	51
300	76
400	101
500	126

The AI Music Generator was built with the aim of providing users with short music for promotions, the average length of Social Media commercials is around 60 seconds, thus, we set the no of elements as 200.

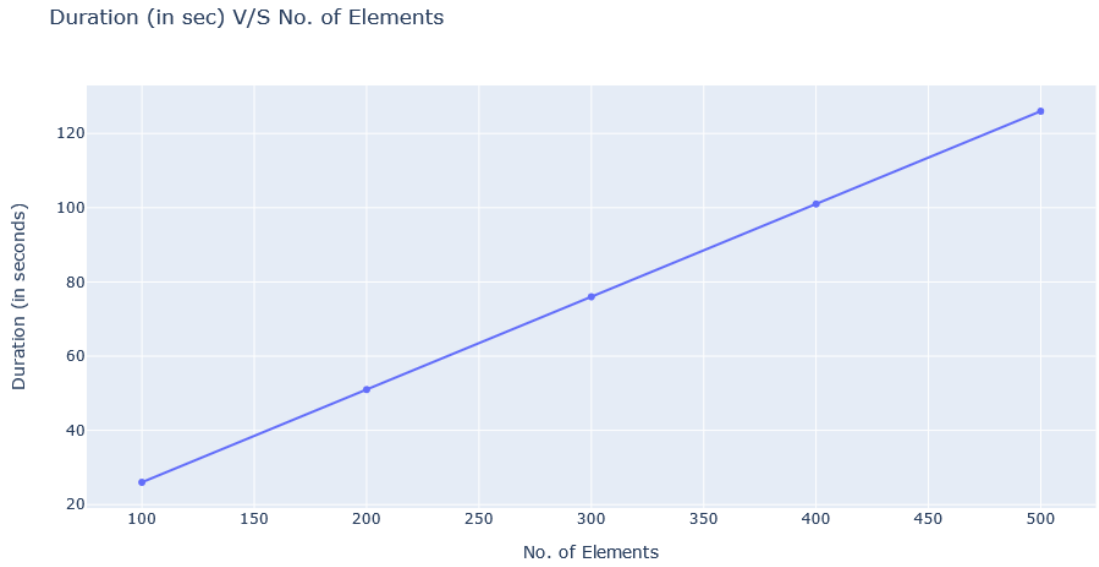

Fig. 11 Duration (in seconds) V/S No. of Elements

D. Varying the Offset to change the duration of Music

The offset is the amount of time, that each note or chord should be played from the predicted output. Analysis shows that the offset is directly proportional to the duration of the generated music.

TABLE IV

DURATION OF GENERATED MUSIC (IN SECONDS) V/S OFFSET

Offset	**Duration (Seconds)**
0.5	51
0.6	61
0.7	71
0.8	81
0.9	91

Changing the offset not only changes the duration of the music, but it also changes how long every single note/chord is played. Thus, increasing or decreasing the Offset beyond a certain value distorts the music. Primary analysis shows, that the suitable range of the Offset value is between 0.5-0.9.

The Music produced when keeping the Offset value as 0.5 is also idea for Social Media promotions & sounds harmonious. Thus, the offset value is set as 0.5.

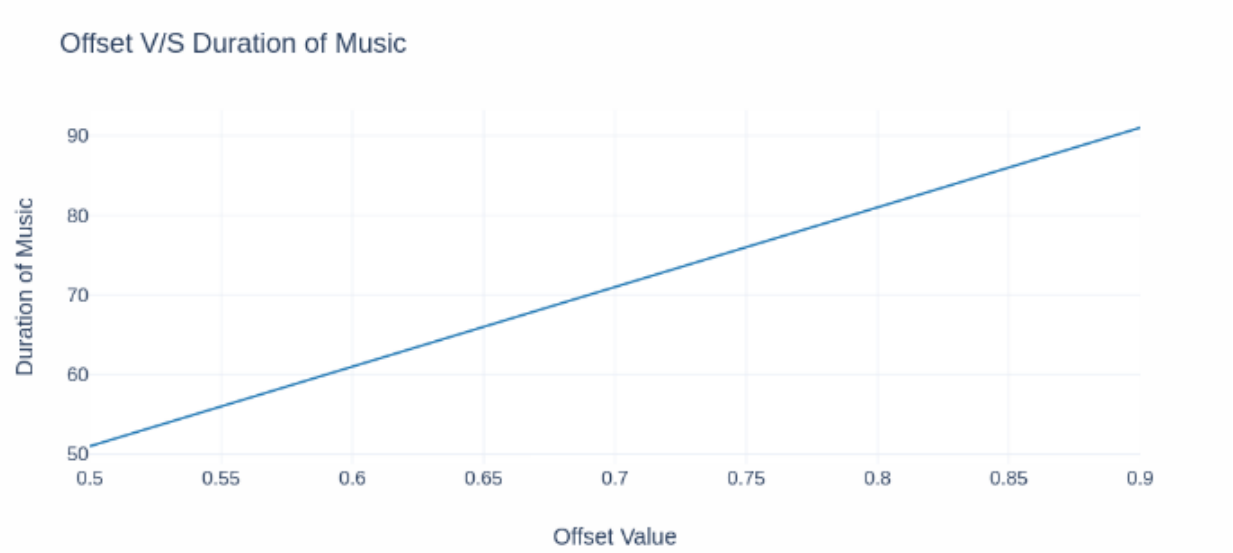

Fig. 12 Duration of Generated Music (in seconds) V/S Offset

E. Summary of Model Generated

TABLE V

SUMMARY OFTHE ML MODEL

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 512)          1052672
_________________________________________________________________
dropout (Dropout)            (None, 100, 512)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 512)          2099200
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 512)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 512)               2099200
_________________________________________________________________
dense (Dense)                (None, 256)               131328
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 201)               51657
=================================================================
Total params: 5,434,057
Trainable params: 5,434,057
Non-trainable params: 0
_________________________________________________________________
```

F. No. of Epochs compared against the Training Time

Epoch is a single iteration where the ML Model processes the entire training set, the more the number of Epochs, the better result is obtained.

TABLE VI

EPOCHS V/S TRAINING TIME (IN MILLI SECONDS)

Epochs	Training Time (in milli seconds)
2	235
10	1175
50	5,875
100	11,750
180	21,150

Since the training phase happens before the Users are allowed to use a particular Model to produce music, thus, The Model has been trained at 180 Epochs. The use of such high Epochs enabled the production of high-quality music.

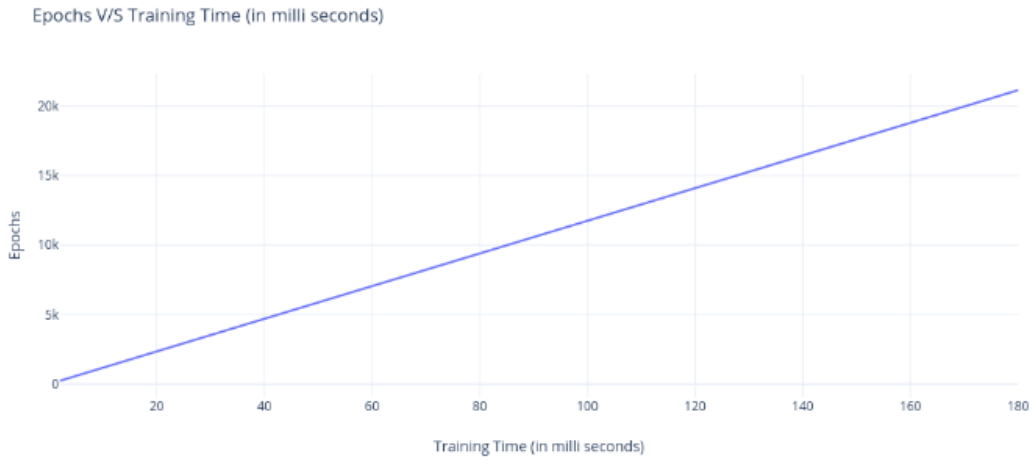

Fig. 13 Epochs V/S Training Time (in milli seconds)

The Optimal settings as specified in every subsection have been used to generate the Music as per the LSTM Algorithm.

G. Word Cloud of the Notes (and Chord) Generated using the Model on the Optimal Settings

Fig. 14 Word Cloud of Generated Music Using Optimal Settings

The Testing section aimed at ensuring that all the individual components of the entire algorithm are optimized to ensure that the Music generated is harmonious and, the User does not have to wait for too long to hear to the generated music either.

Numerical & graphical analysis were used to test all the possible configurations to ensure that all the aspects of the algorithm are verified thoroughly. The result is a set of parameters with their optimized value, which is well suited to the Algorithm.

VIII. RESULTS

A. Comparison of LSTM Model with other models.

Previous work on the various algorithms which aimed to generate music using AI Algorithms shows that a variety of approaches have worked to generate music. The literature survey section of the paper can give insights on the previous work done in the domain. This section aims to compare the previously existing work with the algorithm discussed in this paper. An interesting approach had been taken by Li, H in the paper Piano Automatic Computer Composition by Deep Learning and Blockchain Technology [3]. Here the approach discussed to generate music uses the concept of GRU-RNN Algorithm. The metrics of a few other algorithms have also been discussed in the paper.

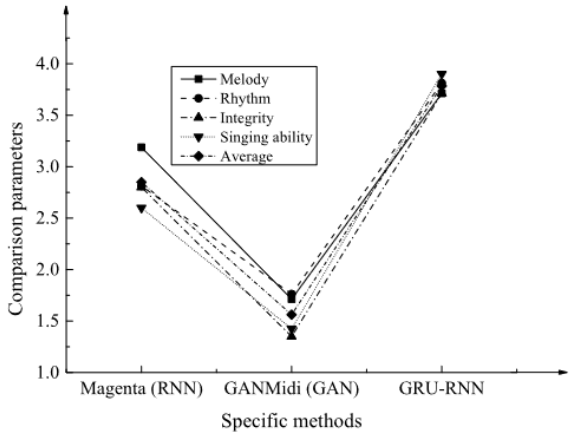

Fig. 15 Comparison of Music Generation Algorithms [3]

LSTM Algorithm used by this paper can be compared to the GRU-RNN Algorithm used by the paper above in terms of various parameters (time, accuracy, etc.) The following graph shows the comparison between GRU-RNN & LSTM Algorithms based on

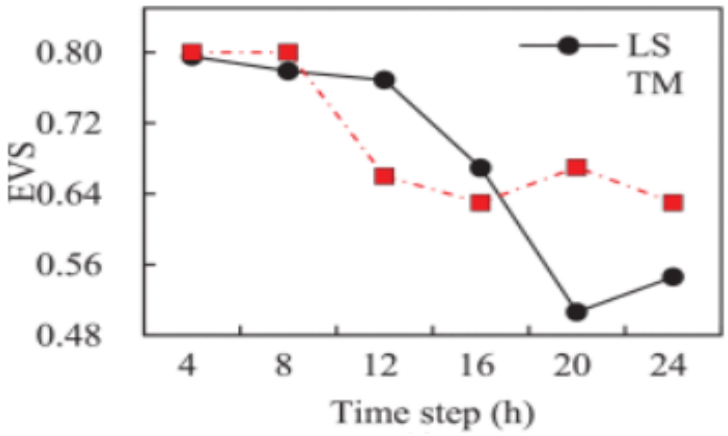

Fig. 16 Explained Variance Score (EVS) Compared against Time Step (h)

The above graph shows clearly, that the time required for each step decreases with increase in EVS, however, the time required by the LSTM Algorithm is lesser than the GRU-RNN Algorithm. LSTM Algorithm also takes slightly higher time to train as compared to GRU, however, it outperforms GRU, especially when the dataset is large.

B. Input & Output

The Users interact with the Website built on Flask. This interaction primarily lets the user understand the use of the website & navigating the user to generate the music. The Music is generated once the user has entered all the relevant details. The user has to select his/her choice of instrument. The algorithm starts processing the request the moment the user has selected the relevant choices.

The parameters other than the instrument of the user's choice are chosen set to their optimum value by default. The Dropout parameter plays an important role in this portion, as the dropout is selected randomly by the algorithm. This ensures that the music generated every time the algorithm is executed is different.

The output generally takes around 60 seconds to get processed with the pre-defined parameters. This time may however depend on the system being used to run the algorithm. The output of the algorithm is music, that has been generated by the algorithm. The music generated on running the algorithm again would be different, due to the Dropout parameter being randomized.

C. Consensus

The optimum values of the algorithm were found using analysis. The previous work suggested that the greater the value of Epochs, the better the output is. The following survey with 148 unbiased candidates validates the same.

The Algorithm was trained on different number of epochs. The models built on different numbers of epochs was then used to produce music. The following are the responses of the candidates from a scale of 1 to 5. Here, 1 denotes that the music produced was Discordant (Unpleasant) and, 5 denotes that the music was Melodious.

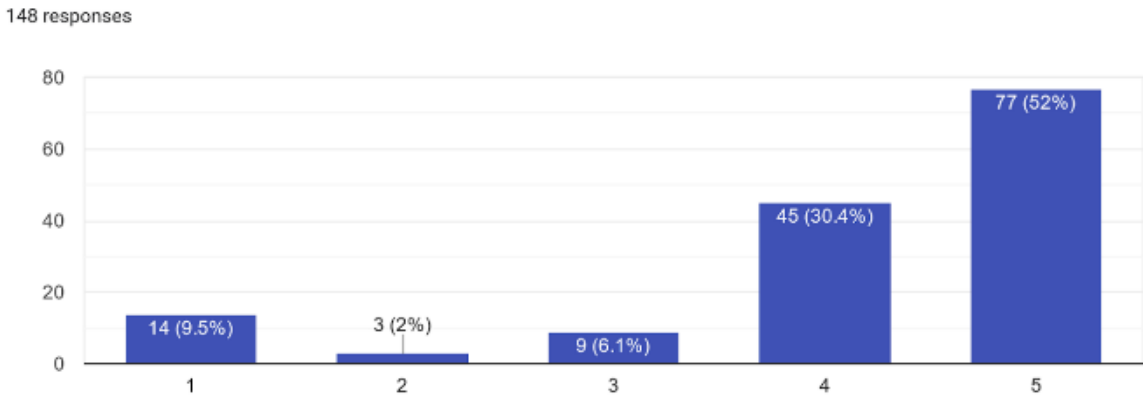

Fig. 17 User's Response to Music trained on 180 epochs

The graph indicates that a large number of people found the music to be melodious. With over 80% of the users selecting the option 4 & 5.

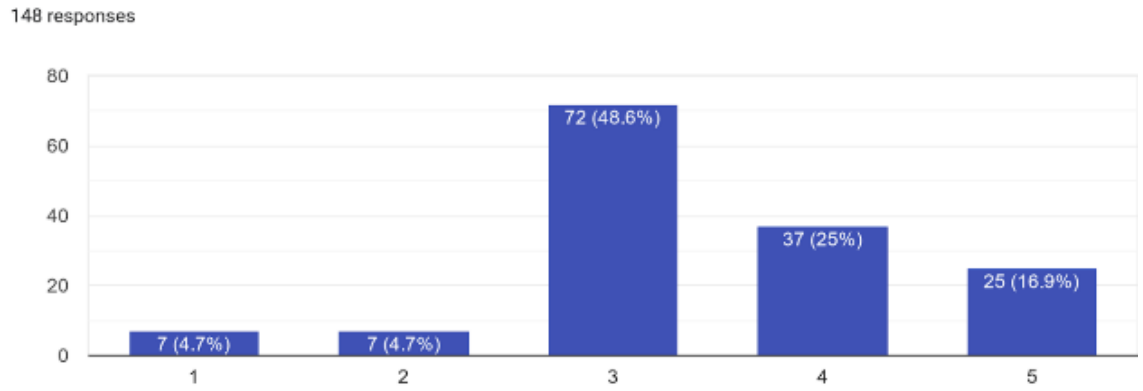

Fig. 18 User's Response to Music trained on 120 epochs

While the users liked the music produced by training the algorithm on 120 epochs, the trend shows that a majority of the users rated the music mediocre (option 3). Thus, the people preferred music trained on 180 epochs.

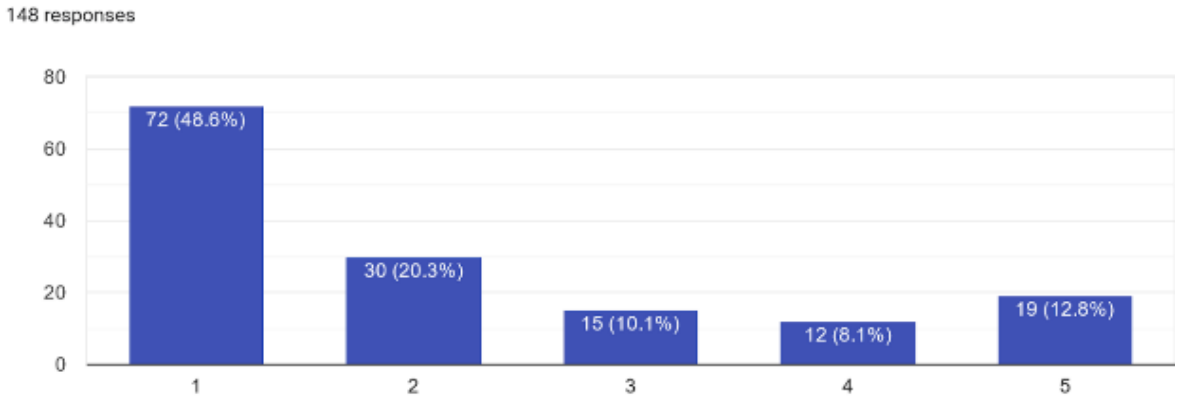

Fig. 19 User's Response to Music trained on 80 epochs

The trend of the graph above clearly shows that the majority of the people found the Music trained on 80 epochs unpleasant. The above results are in line with the concepts of previously researched work, which suggests that increasing the number of epochs leads to better output.

D. Discussion

The rise of internet has led to more services available on the internet due to the rising demand & a larger pool of audience. Music being an identity of a brand is essential for small businesses and start-ups. The production of cost of music however can overwhelming for a lot of companies which have lower amount of financial resources. AI Music Generator can be an asset for such companies. The Service is made more reachable by building a flask app, which can be scaled into a deployable website. The website has a simple user interface, which helps the user to understand how the website works then navigates the users to the webpage where the instrument available are displayed. The user is free to pick the instrument of his/her choice.

The Algorithm generating the music is run, as soon as the user selects the choice of his/her instrument. The Algorithm here producing music is the LSTM algorithm. The Long-short term algorithm remembers patterns of notes or chords of the dataset of music which has been fed into the algorithm. The LSTM Algorithm then produces to generate a sequence of notes or chords based on a variety of parameters. The LSTM Algorithm is pre-trained with a dataset containing music of a particular instrument. The pre-training helps the algorithm to produce the music quicker, as the training time for the model can be as high as 2 days. The algorithm builds a sequence of notes & chords based on the model. The sequence of

notes & chords is fed into the music21 library, which produces the music. The user is redirected to another page, where the generated music can be player. The user can then choose to purchase the digital rights of the produced music.

IX. CONCLUSION

The paper proposed an approach to generate music using the concept of Long Short-Term Memory, based on Neural Networks. The idea of music-generation is further extended, to build a complete platform on web. The concepts of music theory are first understood. The ideas are then expanded to understand how music can be generated from understanding which sequences of notes & chords sound harmonious. Various algorithms are analysed, before choosing the LSTM algorithm, based on recurrent neural networks. Long Short-Term Memory Network can remember patterns of chords which sound harmonious. The ML Module compromises the LSTM algorithm, along with the code for cleaning and extraction of data from the dataset. The ML Module is firstly trained using existing instrumental music.

The users interact with the UI compromising of a web-based platform built on Flask, which is connected to the ML Module. The ML Module, which has been trained using an existing set of music starts running as soon as the user selects their choice of instrument. The Music produced by the LSTM algorithm can then be heard by the user on the webpage. The comparison of LSTM algorithm has been done with a variety of algorithms on various factors. LSTM algorithm performs significantly better than the traditional approaches of generating music. The algorithm has also been tested thoroughly in order to optimize the various parameters of the algorithm. This analysis ensures, that the quality of the song produces & the waiting time are equally optimized. The parameters of the algorithm are further checked, by conducting a survey with music produced using different parameters in the same algorithm. The users are told to rate the music they hear. The most optimum parameters can be concluded using the combination of the above two analysis.

There are certain aspects of the paper, which can be enhanced in the future. The first one being, that the time for which each note or chord plays remains constant. Although the parameter has been optimized, there is still scope for a wider variety of music. The model can also be trained with a greater number of instruments.

REFERENCES

[1] Wang, S. C. (2003). Artificial neural network. In Interdisciplinary computing in java programming (pp. 81-100). Springer, Boston, MA.

[2] Gers, F. A, Schmidhuber J, & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM

[3] Li, H. (2020). Piano Automatic Computer Composition by Deep Learning and Blockchain Technology. *IEEE Access*, *8*, 188951-188958.

[4] Ycart, A., & Benetos, E. (2020). Learning and Evaluation Methodologies for Polyphonic Music Sequence Prediction With LSTMs. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 1328-1341.

[5] Sigtia, S., Benetos, E., & Dixon, S. (2016). An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(5), 927-939

[6] Han, Y., Kim, J., & Lee, K. (2016). Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(1), 208-221.

[7] Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2018). The effects of noisy labels on deep convolutional neural networks for music tagging. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(2), 139-149.

[8] Zhang, N. (2020). Learning adversarial transformer for symbolic music generation. *IEEE Transactions on Neural Networks and Learning Systems*.

[9] Hutchings, P. E., & McCormack, J. (2019). Adaptive music composition for games. *IEEE Transactions on Games*, *12*(3), 270-280.

[10] Kolozali, Ş., Barthet, M., Fazekas, G., & Sandler, M. (2013). Automatic ontology generation for musical instruments based on audio analysis. *IEEE transactions on audio, speech, and language processing*, *21*(10), 2207-2220.

[11] Qi, Y., Liu, Y., & Sun, Q. (2019). Music-Driven Dance Generation. *IEEE Access*, *7*, 166540-166550.

[12] Feng, J., Chai, Y., & Xu, C. (2021). A Novel Neural Network to Nonlinear Complex-variable Constrained Nonconvex Optimization. *Journal of the Franklin Institute*.

[13] Elmer, S. (2016). Relationships between music training, neural networks, and speech processing. *International Journal of Psychophysiology*, *100*(108), 46.

[14] Rodgers, W., Yeung, F., Odindo, C., & Degbey, W. Y. (2021). Artificial intelligence-driven music biometrics influencing customers' retail buying behavior. *Journal of Business Research*, *126*, 401-414.

[15] Schiller, I. S., Morsomme, D., & Remacle, A. (2018). Voice use among music theory teachers: a voice dosimetry and self-assessment study. *Journal of Voice*, *32*(5), 578-584.

[16] Beim Graben, P., & Blutner, R. (2019). Quantum approaches to music cognition. *Journal of Mathematical Psychology*, *91*, 38-50.

[17] Liu, F., Zhang, L., & Jin, Z. (2020). Modeling programs hierarchically with stack-augmented LSTM. *Journal of Systems and Software*, *164*, 110547.

[18] Herremans, D., & Chew, E. (2017). MorpheuS: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, *10*(4), 510-523.

[19] Sturm, B. L., Ben-Tal, O., Monaghan, Ú., Collins, N., Herremans, D., Chew, E., ... & Pachet, F. (2019). Machine learning research that matters for music creation: A case study. *Journal of New Music Research*, *48*(1), 36-55.

[20] Gao, S., Huang, Y., Zhang, S., Han, J., Wang, G., Zhang, M., & Lin, Q. (2020). Short-term runoff prediction with GRU and LSTM networks without requiring time step optimization during sample generation. *Journal of Hydrology*, *589*, 125188.

[21] Wang, S., Zhao, J., Shao, C., Dong, C. D., & Yin, C. (2020). Truck Traffic Flow Prediction Based on LSTM and GRU Methods With Sampled GPS Data. *IEEE Access*, *8*, 208158-208169.

[22] Hizlisoy, S., Yildirim, S., & Tufekci, Z. (2020). Music emotion recognition using convolutional long short term memory deep neural networks. *Engineering Science and Technology, an International Journal*.

[23] Jiao, M., Wang, D., & Qiu, J. (2020). A GRU-RNN based momentum optimized algorithm for SOC estimation. *Journal of Power Sources*, *459*, 228051.

[24] Dong, S., & Ni, N. (2021). A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, *435*, 110242.

[25] Mukherjee, H., Dhar, A., Ghosh, M., Obaidullah, S. M., Santosh, K. C., Phadikar, S., & Roy, K. (2020). Music chord inversion shape identification with LSTM-RNN. *Procedia Computer Science*, *167*, 607-615.

[26] Wu, J., Liu, X., Hu, X., & Zhu, J. (2020). PopMNet: Generating structured pop music melodies using neural networks. *Artificial Intelligence*, *286*, 103303.

[27] Mittal, S., & Umesh, S. (2020). A Survey on Hardware Accelerators and Optimization Techniques for RNNs. *Journal of Systems Architecture*, 101839.

[28] Vijayaprabakaran, K., & Sathiyamurthy, K. (2020). Towards activation function search for long short-term model network: A differential evolution based approach. *Journal of King Saud University-Computer and Information Sciences*.

[29] Bagui, S. C., Mehra, K. L., & Vaughn, B. K. (1997). An M-stage version of the k-RNN rule in statistical discrimination. *Journal of statistical planning and inference*, *65*(2), 323-333.

[30] Wu, J., Hu, C., Wang, Y., Hu, X., & Zhu, J. (2019). A hierarchical recurrent neural network for symbolic melody generation. *IEEE transactions on cybernetics*, *50*(6), 2749-2757.

FORMAT: The Institute of Electrical and Electronics Engineers (IEEE)