iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** IV **Month of publication:** April 2023

DOI: https://doi.org/10.22214/ijraset.2023.51184

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

musicAL.GO - Algorithmic Music Generation

Prajkta Ghole[1], Jash Ratanghayara[2], Rushikesh Patil[3], Dr. Prof. Vipul Dalal[4]
Department of Information Technology,Vidyalankar Institute of Technology Mumbai, India

Abstract: Composing music is a very interesting challenge that tests the composer's creative capacity, whether it is a human or a computer. Although there have been many arguments on the matter, almost all of music is some regurgitation or alteration of a sonic idea created before. Thus, with enough data and the correct algorithm, machine learning and deep learning algorithms and techniques should be able to make music that would sound human. This paper outlines approaches to music composition through Neural Network models, specifically char-RNN and although there were some mixed results by the model, it is evident that musical ideas can be gleaned from these algorithms in hopes of making a new piece of music. Based on this principle of algorithmically generating new music, this project aims at creating a web application for music lovers providing a repository of copyright free musical pieces generated artificially, enabled with numerous other features and functionalities.

I. INTRODUCTION

Music is deeply embedded in our everyday lives from listening to the radio to YouTube music videos. With everyone having a distinct music taste, the area of music is only expanding. Everyday new songs are being created and the art of music holds the interests of many all across the world from different countries and cultures.

In today's world, it's a myth that you need to be a music expert to generate music. Even a person who likes music can produce good quality music. We all like to listen to music and if it is possible to generate music automatically then it will prove to be a new revolution in the world of music industry. Until very recently, all music generation was done manually by means of analogue signals. In recent years though music production is done through technology, assisted by humans. The task that has been accomplished in the project is the construction of generative neural network architectures that can efficiently portray the complex details of harmony and melody without the need for human interruption. A brief summary of the precise details of music and its mechanisms has been provided in the project with appropriate citations where required. The primary objective of the project was to devise an algorithm that can be used to create musical notes utilizing Long Short-Term Memory (LSTM) networks in Recurrent Neural Networks (RNNs). The output data obtained is in ABC notation which is then converted into MP3. To train the model we have chosen to work on ABC notations. ABC notation is basically one of the ways to represent music. It consists of two parts. First part represents the meta data which comprises various characteristics of tines such as index, time signature, default note length and type of tune. The second part represents the actual tune which is nothing but a sequence of characters. The devised algorithm learns the sequences of monophonic musical notes over three single layered LSTM-network.

In Section II, Literature Review of the paper is discussed wherein, technical research papers and some existing systems were studied. In Section III, Proposed Approach is elaborated. In Section IV, the dataset and the implementation part of the system are discussed in detail. In Section V, the results are presented and discussed. In Section VI, future scope of the project citing various ways of project application is mentioned. In Section VII, conclusions are stated. In Section VIII, references are quoted.

II. LITERATURE REVIEW

Related Work - Music generation has been at the epicenter of attention of members of the research community and has thus been studied upon a lot. Many who did try to generate music have done so using different approaches. Hence there exists numerous ways in which music can be generated and an amalgamation of such approaches can be used to create and design a new yet competent model. These approaches have been divided into two main categories – Traditional and Autonomous. Traditional approach uses algorithms working on already defined functions to make music, whereas Autonomous Model learn from the prior iterations of the notations and then generates new ones. Algebra founded upon the usage of the tree structure to enforce grammar constitutes one of the earlier attempts. Markov chains can be used to design such a model. Many models and approaches have been documented in the field of artificial intelligence soon after the field experienced a massive boom. Such models include probabilistic models which use variants of RNN, namely Char RNNs, Anticipation RNNs. One method that is actively being used to generate musical notes is the Generative adversarial networks (GANs) which contains two neural networks – discriminator network and the generator network.

The generator network and the discriminator network work in tandem to evaluate authenticity of the generated data against the

original dataset. Research studies reveal that LSTM outclasses the GAN in terms of fixating on certain sequences, that is, LSTM is superior when it comes to uncovering specific patterns and then recycling them throughout the course of the output sequence. The models based upon LSTM were able to get out of certain note loops and shift into other notes [9]. When it comes to GAN, it can only pick up on basic concepts, albeit better, and exhibits shorter training time.

- In [12], they propose an algorithm which can be used to generate musical notes using Recurrent Neural Networks (RNN), principally Long Short-Term Memory (LSTM) networks. The model is capable to recall the previous details of the dataset and generate a polyphonic music using a single layered LSTM model, proficient enough to learn harmonic and melodic note sequence from MIDI files of Pop music.
- In [13], they created a multilayer Long-Short Term Memory (LSTM) Recurrent Neural Network (RNN) and feed-forward network, based on acquired dataset; and a LSTM based Encoder-Decoder architecture as baseline models. The work so far did not fulfil the set goal of generating a 60-second-long sequence of polyphonic music. They discussed the interpretation of the limitations of the models that was used. There is a need of further refinement before being able to generate actual musical sequences.
- In [14], a combination of a new architecture of an artificial neural network and variational autoencoder supported by history, with filtering heuristics allows generating pseudo-live acoustically pleasing and melodically diverse music. This is the first application of VRASH to music generation. It provides a good balance between the global and local structure of the track. The proposed structure is relatively easy to implement and train. It also allows to control the style of the output and generate tracks corresponding to the given parameters
- In [15], they created a new metric to assess the quality of generated music and use this measure to evaluate the outputs of a truly generative model based on Variational Autoencoders that will apply to automated music composition. They used it to automatically and systematically fine-tune the generative model's parameters and architectures for optimizing the musical out- puts in terms of proximity to a specific musical style.

III. PROPOSED APPROACH

A. *Proposed System*:

The primary objective of the project was to devise an algorithm that can be used to create musical notes utilizing Long Short-Term Memory (LSTM) networks in Recurrent Neural Networks (RNNs). The output data obtained is in ABC notation which is then converted into MP3. To train the model we have chosen to work on ABC notations. ABC notation is basically one of the ways to represent music. It consists of two parts. First part represents the meta data which comprises various characteristics of tines such as index, time signature, default note length and type of tune. The second part represents the actual tune which is nothing but a sequence of characters. The devised algorithm learns the sequences of monophonic musical notes over three single layered LSTM network. The proposed system allows the user to play music and recommends music based on what the user likes. Music recommender system is a system which learns from the users past listening history and recommends them songs which they would probably like to hear in future. We haveimplemented various algorithms to try to build an effective recommender system. We firstly implemented popularity-based model which was quite simple and intuitive. Collaborative filtering algorithms which predict (filtering) taste of a user by collecting preferences and tastes from many other users (collaborating) is also implemented. We have also done experiments on content-based models, based on latent factors and metadata.

B. *Proposed Methodology*

1) *Objectives and Technical Challenges:* The first problem faced when dealing with music is that of its representation. Signals, MIDI, notations, etc. are all possible representations. Due to the higher efficacy of notations for the task at hand, the model used in this paper is fed, processes and outputs using ABC notations. The ABC notation uses 7 letters (A to G) with other symbols representing features such as – flat, sharp, note length, key, etc. to represent the given notes. The generation of output also posed a challenge as in our model, the output has to be human compatible, that is to say that the output should be understandable by humans. The ABC notation-based sequence from the output is then converted into MIDI format which in turn is converted into MP3 for easy listening.
2) *Design:* RNNs fall victim to the vanishing/exploding gradient problem due to their use of back-propagation, to remedy this we have employed the use of LSTMs. At each timestep of the RNN, the individual LSTM cell is fed a value, the cell then calculates the hidden vector and outputs is to the next timesteps. The current input, and the previous hidden and memory states are given as input to the cell. Similarly, the current hidden state and the current memory state are the outputs. Taking the current timestep as t, the current hidden vector ht is found using the current input at and the previous timestep's hidden vector ht-1. This is how RNNs process data sequentially.

3) *DataSet:* The dataset is obtained from http://abc.sourceforge.net/NMD. It says "ABC version of the Nottingham Music Database" which contains over 1000 folk tunes stored in a special text format. Of course, it takes a lot of time to train the model with larger data like 1000 tunes. So, we used the jigs dataset which contains about 340 tunes. We get a txt file with multiple tunes here. We simply copy and paste into a txt file as input.txt. Each tune is having a meta data section and music section.
4) *Data Processing:* The model is designed to interpret the musical notes in the form of 87 unique characters present in a dictionary, using integer encoding. The dataset is hot-encoded as well, to convert the labels into binary vectors. This is then fed into the LSTM units in batches. The specification of the batches used here are as follows: Batch Size = 16, Sequence Length=64.

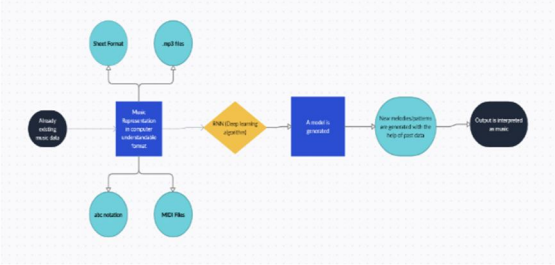


Figure 1 – Proposed Flow Chart

IV. EXPERIMENTAL SETUP

A. *Char-RNN Model (High-Level Overview):*

Since our music is a sequence of characters, therefore our obvious choice will be RNNs.

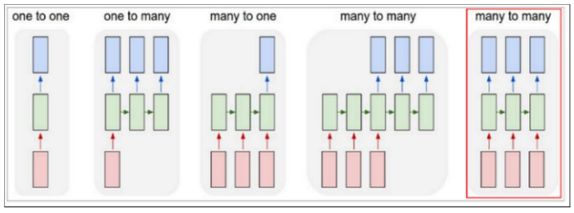


Figure 2 – Comparison between various types of RNNs

There is a special type of RNN called "Char-RNN". We will be using many to many RNN. Here, we will feed the RNN with our characters of the sequence one by one and it will out the next character in the sequence.

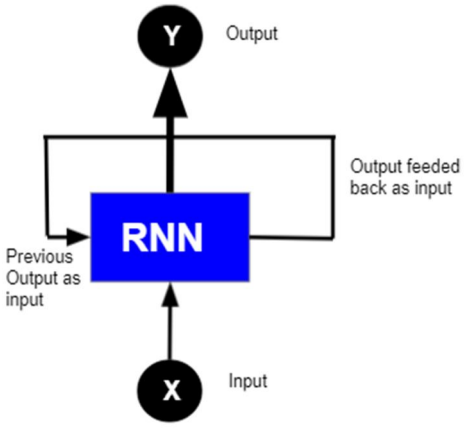


Figure 3 – Context Sensing and feedback mechanism of RNNs

This will then allow us to generate a new tune one character at a time.
As a working example shown in Figure 4, suppose we only had a vocabulary of four possible letters "helo", and wanted to train an RNN on the training sequence "hello".

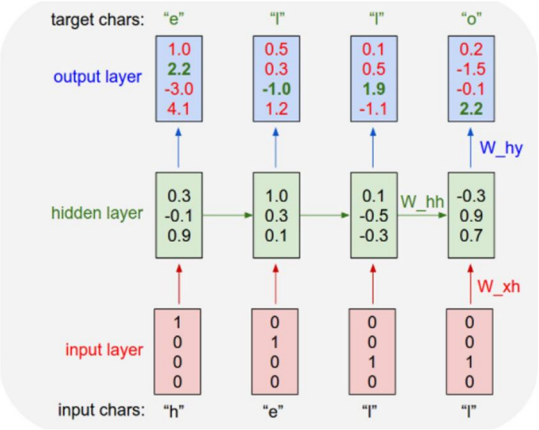


Figure 4 – Example sequence generated by char-RNN

This training sequence is in fact a source of 4 separate training examples:

1) The probability of "e" should be likely given the context of "h".
2) "l" should be likely in the context of "he".
3) "l" should also be likely given the context of "hel".
4) Finally, "o" should be likely given the context of "hell".
5) We will encode each character into a vector using one-hot encoding(i.e. all zero except for a single one at the index of the character in the vocabulary) and will feed them into RNN one at a time with the step function.
6) We will then observe a sequence of 4-dimensional output vectors (one dimension per character), which we interpret as the confidence of the RNN currently assigns to each character coming next in the sequence.
7) want the green numbers to be high and the red numbers low in the output layer, giving us the target characters.
8) Remember that as this many to many models, the number of inputs is equal to number of outputs.

B. Data

The first part is obtaining data itself. The second part is pre-processing the data and generating each of the individual batches which we will use along with its construction as we will be applying a variation of the Batch-SGD while training the Char-RNN.

C. Data Obtaining

Refer: http://abc.sourceforge.net/NMD.It says "ABC version of the Nottingham Music Database" which contains over 1000 folk tunes stored in a special text format. Of course, it takes a lot of time to train the model with larger data like 1000 tunes. So, we used the jigs dataset which contains about 340 tunes. We get a txt file with multiple tunes here. Simply copy and paste into a txt file as input.txt. Each tune is having a meta data section and music section.

D. Data Preprocessing

We want to preprocess the input.txt file into such a format that we can feed it into the RNN because the way we build our dataset will impact the model heavily and RNNs can be tricky. Batch Size=16, Sequence length=64, Total length of characters in input.txt file = 129, 665, No of unique characters = 86

char_to_idx = {ch: i for (i, ch) in enumerate(sorted(list(set(text))))}

Here, in char_to_idx, char-to-idx is converting every character to a index or numerical value where ch(character) is the key and index or numerical value, which is similar to

{"\n": 0, " ": 1, "!": 2, "\"": 3, "#": 4, "%": 5, "&": 6, "'": 7, "(": 8, ")": 9, "+": 10, ",": 11, "-": 12, ".": 13, "/": 14, "0": 15, "1": 16, "2": 17, "3": 18, "4": 19, "5": 20, "6": 21, "7": 22, "8": 23, "9": 24, ":": 25, "=": 26, "?": 27, "A": 28, "B": 29, "C": 30, "D": 31, "E": 32, "F": 33, "G": 34, "H": 35, "I": 36, "J": 37, "K": 38, "L": 39, "M": 40, "N": 41, "O": 42, "P": 43, "Q": 44, "R": 45, "S": 46, "T": 47, "U": 48, "V": 49, "W": 50, "X": 51, "Y": 52, "[": 53, "\\": 54, "]": 55, "^": 56, "_": 57, "a": 58, "b": 59, "c": 60, "d": 61, "e": 62, "f": 63, "g": 64, "h": 65, "i": 66, "j": 67, "k": 68, "l": 69, "m": 70, "n": 71, "o": 72, "p": 73, "q": 74, "r": 75, "s": 76, "t": 77, "u": 78, "v": 79, "w": 80, "x": 81, "y": 82, "z": 83, "|": 84, "~": 85}

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

E. Many to Many RNN

In the hidden layer, instead of having single lstm, we can have multiple lstm for production purposes. In reality, we can have setup like the below figure.

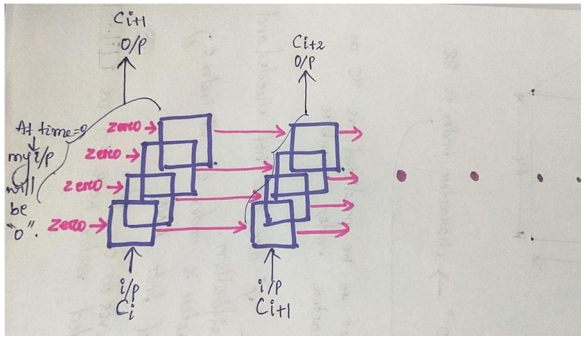


Figure 5 – Many to Many RNNs

Let's say, we have four lstms in the hidden layer. I have an input "Ci" at time "t" in the input layer. Now, all four of the lstms will together generate an output for me, which should be ideally "Ci+1" character. The output from the first-time step will go to the next time step, where my input is "Ci+1". Again, there will be four lstms working together to give me output "Ci+2". The process will continue so on. Each of the lstm unit will learn different aspect of our character.

F. Model Architecture & Training

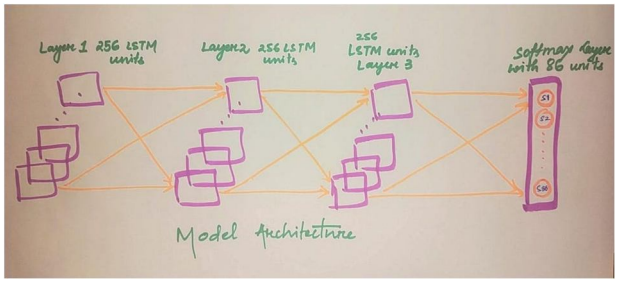


Figure 6 – Model Architecture

X is a matrix of (BATCH_SIZE, SEQ_LENGTH) = (16,64). Y is a 3D tensor of (BATCH_SIZE, SEQ_LENGTH, vocab_size) = (16,64,86). The vocab size is considered because of one-hot encoding. After embedding, (BACTH_SIZE, SEQ_LENGTH, embedding_dim) = (16,64,512). We encoded "Y" as one hot encoded because we will be applying softmax on top of it. Now, we want to predict the next character which should be one of the 86 unique characters. So, it's a multi-class classification problem. Therefore, our last layer is the softmax layer of 86 activations. So, we will generate each of the batches and train them. For every training epoch, we will print the categorical cross-entropy loss and accuracy. Here is the summary of the model. So, we have 1,904,214 total parameters.

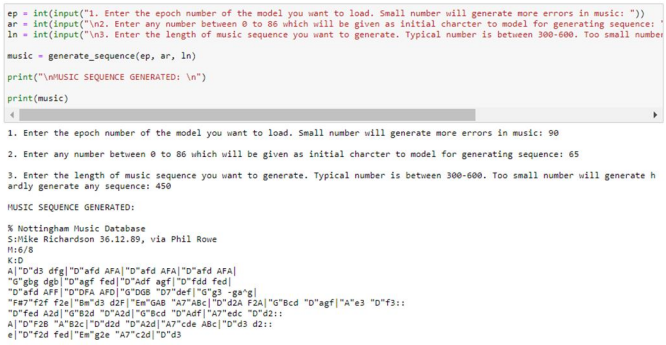
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (16, 64, 512)             44032
_________________________________________________________________
lstm (LSTM)                  (16, 64, 256)             787456
_________________________________________________________________
dropout (Dropout)            (16, 64, 256)             0
_________________________________________________________________
lstm_1 (LSTM)                (16, 64, 256)             525312
_________________________________________________________________
dropout_1 (Dropout)          (16, 64, 256)             0
_________________________________________________________________
lstm_2 (LSTM)                (16, 64, 256)             525312
_________________________________________________________________
dropout_2 (Dropout)          (16, 64, 256)             0
_________________________________________________________________
time_distributed (TimeDistri (16, 64, 86)              22102
_________________________________________________________________
activation (Activation)      (16, 64, 86)              0
=================================================================
Total params: 1,904,214
Trainable params: 1,904,214
Non-trainable params: 0
_________________________________________________________________
```

Figure 7 – Model Description & Summary

As we are having so many parameters, so we are using dropouts with a keep probability of 0.2.By the time we reach 100 epochs while training, roughly around 90% + times, the model is able to predict what the next character is. So, our model is doing a pretty good job.At the end of 10 epochs, we are storing the weights of the model. We will use these weights to reconstruct the model and predict.

V. RESULTS

In order to make prediction, we will give any of the 87 unique characters as input to our model, and it will generate 87 probability values through the softmax layer. From these returned 87 probability values, we will choose the next character probabilistically and finally, we will again feed the chosen character back to the model and so on. We will keep concatenating the output characters and generate music of the desired length. The typical number is between 300–600. A too small number will hardly generate any sequence. We will get errors, when we try to predict music using weights of smaller epochs.



```
ep = int(input("1. Enter the epoch number of the model you want to load. Small number will generate more errors in music: "))
ar = int(input("\n2. Enter any number between 0 to 86 which will be given as initial charcter to model for generating sequence: "
ln = int(input("\n3. Enter the length of music sequence you want to generate. Typical number is between 300-600. Too small numbe

music = generate_sequence(ep, ar, ln)

print("\nMUSIC SEQUENCE GENERATED: \n")

print(music)
```

```
1. Enter the epoch number of the model you want to load. Small number will generate more errors in music: 90

2. Enter any number between 0 to 86 which will be given as initial charcter to model for generating sequence: 65

3. Enter the length of music sequence you want to generate. Typical number is between 300-600. Too small number will generate h
ardly generate any sequence: 450

MUSIC SEQUENCE GENERATED:

% Nottingham Music Database
S:Mike Richardson 36.12.89, via Phil Rowe
M:6/8
K:D
A|"D"d3 dfg|"D"afd AFA|"D"afd AFA|"D"afd AFA|
"G"gbg dgb|"D"agf fed|"D"Adf agf|"D"fdd fed|
"D"afd AFF|"D"DFA AFD|"G"DGB "D7"def|"G"g3 -ga^g|
"F#7"f2f f2e|"Bm"d3 d2F|"Em"GAB "A7"ABc|"D"d2A F2A|"G"Bcd "D"agf|"A"e3 "D"f3::
"D"fed A2d|"G"B2d "D"A2d|"G"Bcd "D"Adf|"A7"edc "D"d2::
A|"D"f2B "A"B2c|"D"d2d "D"A2d|"A7"cde ABc|"D"d3 d2::
e|"D"f2d fed|"Em"g2e "A7"c2d|"D"d3
```

Figure 8 – New Music Sequence Generation Results

VI. FUTURE SCOPE

Well, we got pretty good results, but we can improve our model by training it with more tunes of multi-instruments. Here, we trained the model with just 350 tunes. So, we can expose our model to more instruments and varieties of musical tunes, which will result in more melodious tunes with varieties. We can add a method, which can handle unknown notes in the music by filtering unknown notes and replacing them with known notes. As of now, my model is trained on a single instrument, so the music generated is of only one instrument. By adding the above method and training the model on large varieties of data while increasing the number of classes, we can generate more robust quality and melodious music.

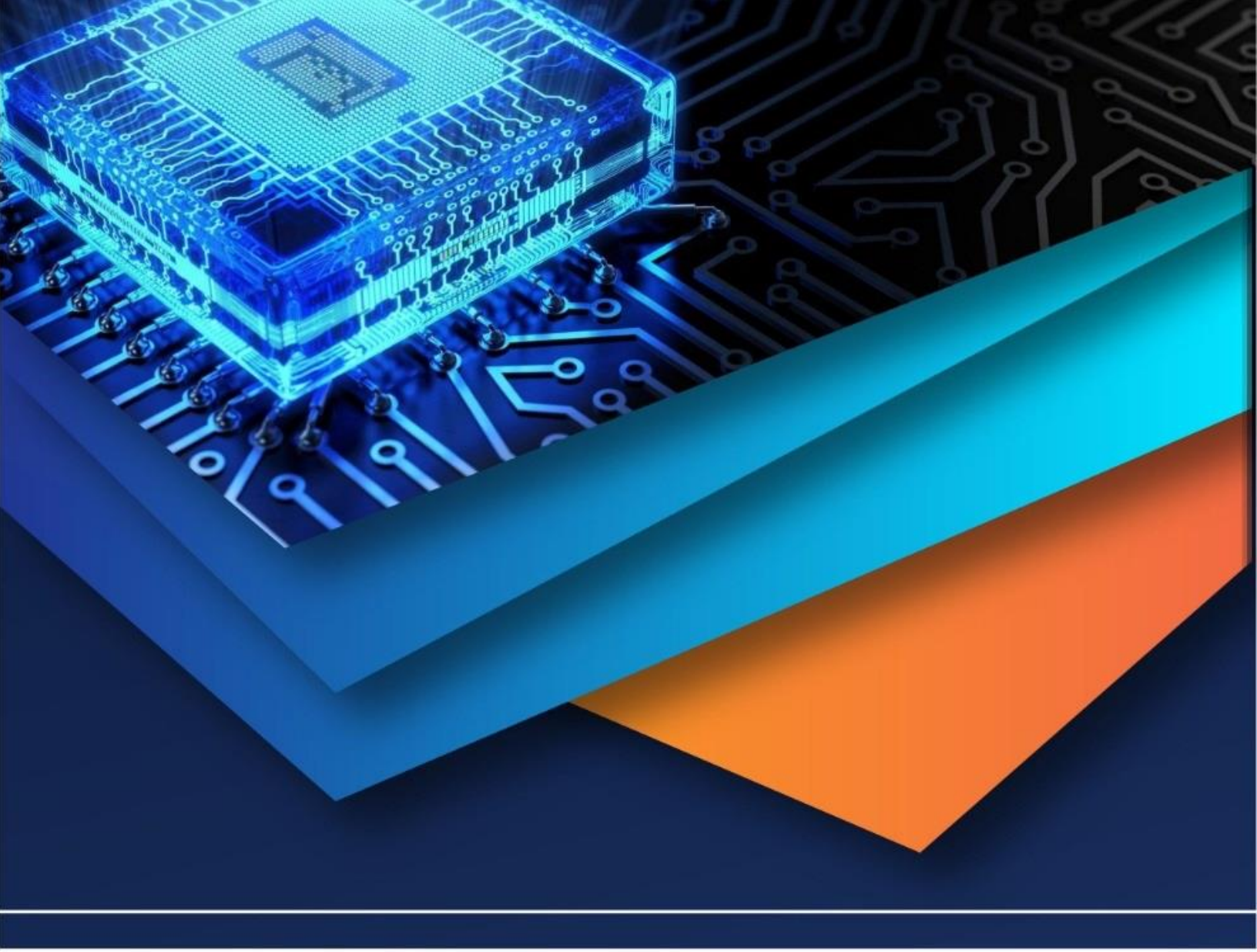
VII. CONCLUSION

The paper introduces an attempt to generate music algorithmically, as music is nothing but a sequence of events, with the help of deep learning techniques. The paper has thus proved the efficiency of a char RNN model at producing a musical sequence that matches the dataset in terms of the sequence's grammatical coherence. The final output, once converted into MIDI, is surreal to most casual listeners' ears. Most of the audience couldn't identify any discrepancies with the sequence generated.

REFERENCES

[1] Manuel Alfonseca," A simple genetic algorithm for music generation by means of algorithmic information theory" IEEE Congress on Evolutionary Computation, Singapore
[2] C. Lipton," A Critical Review of Recurrent Neural Networks", 2015
[3] Joshi," A Comparative Analysis of Algorithmic Music Generation on GPUs and FPGAs" 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)
[4] a. H. J. Drewes," An algebra for tree-based music generation", 2007
[5] W. Van Der Merwe," Music generation with Markov models", 2011
[6] Boulanger-Lewandowski," Modelling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription", 2012
[7] N. Hadjeres," Interactive Music Generation with Positional Constraints using Anticipation-RNNs", 2017
[8] Olof Mogren," C-RNN-GAN: Continuous recurrent neural networks with adversarial training", 2016
[9] Nikhil Kotecha," Generating Music using an LSTM Network",2018
[10] Chen, C-CJ, and Risto Miikkulainen." Creating melodies with evolving recurrent neural networks." In Neural Networks, 2001
[11] McFee, B., Bertin Mahieux, T. Ellis, D. P. Lanck- riet, G. R. (2012, April). The million- song dataset challenge. In Proceedings of the 21st international conference companion on World Wide Web (pp. 909916).ACM.

[12] S. Mangal, R. Modak, and P. Joshi, "LSTM Based Music Generation System," in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 2-6.
[13] M. Docevski, E. Zdravevski, P. Lameski, and A. Kulakov, "Towards Music Generation With Deep Learning Algorithms," in 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2018, pp. 2-4.
[14] I.P. Yamshchikov and A. Tikhonov, "Music Generation with Variational Recurrent Autoencoder Supported by History," in 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2020, pp. 4-9.
[15] R. Sabathe, E. Coutinho and B. Schuller, "Deep Recurrent Music Writer: Memory-enhanced Variational Autoencoder-based Musical Score Composition and an Objective Measure," 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 2021, pp. 2-4
[16] https://folkrnn.org/
[17] https://soundcloud.com/trivedigaurav/char-rnn-composes-long-composition/
[18] https://soundcloud.com/sigur-ur-sk-li/neuralnet-music-1
[19] https://www.researchgate.net/publication/351708912_Music_Generation_using_Deep_Learning



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY