International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org || PP. 108-113

Music Generation using Deep Learning and Implementation on Piano

Niharika S. J. , Nityashree N. , Rachana R. Shankar , Sanjana Aralelemath, K. N. Pushpalatha
[1](*Electronics and Communication Engineering, Dayananda Sagar College of Engineering, India*)
[2](*Electronics and Communication Engineering, Dayananda Sagar College of Engineering, India*)
[3](*Electronics and Communication Engineering, Dayananda Sagar College of Engineering, India*)
[4](*Electronics and Communication Engineering, Dayananda Sagar College of Engineering, India*)
[5](*Electronics and Communication Engineering, Dayananda Sagar College of Engineering, India*)

Abstract: *Music composition is an art form that has been confined to only human beings for many civilizations. Automation of an art form is the first step toward building creative Artificial Intelligence. Musical instruments are made in such a way that humans can use them to play music. But automation of musical instruments paves the way for the creative automated behavior of machines. This paper presents a step-by-step procedure for generating music using the Long Short Term Memory Recurrent Neural Network (LSTMRNN) and the designing of a circuit that enables the automatic pressing of the keys of a piano (or keyboard) to play the generated music. This work will be critical in designing a system that accurately mimics human creativity and actions.*

Keywords *- arduino, deep learning, LSTM, piano, RNN, solenoids*

I. Introduction

Music is a form of art that was initially thought to be unique to humans. People generally assess the advance of artificial intelligence based on its performance on laborious tasks like lifting, assembling, etc. But to achieve the level of Artificial General Intelligence (AGI), it's necessary to make AI creative. By learning to generate music through deep learning models, we can make AI more human-like.

In this paper, we present the use of the LSTM model architecture to generate music. The task of music generation can be achieved through two methods - (a) Traditional and (b) Autonomous [8]. In the former approach, pre-defined functions for producing music are used, whereas, in the latter approach, music is produced by learning patterns in a dataset. We take the autonomous approach using a Classical Music MIDI dataset that contains music compositions of various famous pianists. We have chosen monophonic music for this paper, that is, the music used for training the model only has one instrument.

Recurrent Neural Networks (RNN) are used for sequential learning tasks, and thus, have also been adopted for generating music. Due to their vanishing/exploding gradient problem, LSTMs have become the go-to solution for applications that require learning long-term dependencies.

This work can be divided into two parts - (i) Music Generation using deep learning and (ii) Implementation of the generated music on a keyboard. In the first part, the music is extracted, explored, and pre-processed before using it to train the LSTM model architecture consisting of two LSTM layers. The model's evaluation metric is done using the loss value and the music itself is rated subjectively by a group of undergraduate students. In the second section, the generated music is fed to arduino after some processing and then the microcontroller is used to send command signals to the linear actuators which in turn press the keys of the keyboard and produce music.

II. Related Work

One of the earliest papers on deep learning-generated music, written by Chen et al [1], describes the generation of one music with only one melody and no harmony. This kind of music did not have a global structure and could be improved by adding rhythm to it and utilizing all types of notes including dotted notes, longer chords, and rests. Liu et al. [2] address this issue but failed to overcome the challenges posed by Chen et al.

Eck et al. [3] use two different LSTM networks. One network learns chord structure and local note structure and the other network learns long-term dependencies so that the melody can be learned and retained throughout the piece. Hence the generated music never diverges from the original chord progression melody.

Huang et al. [4] build a generative model from a deep neural network architecture to generate musicthat has both harmony and melody and is passable as music composed by humans.

The complex dynamics of the embodied interactions between a human and a piano were investigated byScimeca et al. [5]. This helps to gain insights into the nature of humans' physical dexterity and adaptability. For automatic music, generation, the dynamic interactions become particularly crucial for delicate expressions, often present in musical pieces, which is the primary focus of the paper. Malloch et al. [6] discuss some digital musical instrument (DMI) design principles in the context of different goals and constraints. It shows, through several examples, that a variety of conditions can motivate design choices for sensor interface and mappings, such as robustness and reliability. Finally, the paper that is closely related to our project is Hayashi et al [7]. The paper describes the development of a superior automated piano that is capable of reproducing the desired performance with expressive soft tones.

III. Methodology

This part of the paper puts forward the steps for the generation process from downloading the dataset to comparing the results. The music processing procedure involves seven major steps. While we present a novel model architecture, the exploration and preprocessing of the dataset take shape mainly from [9]

Part 1: Music Generation

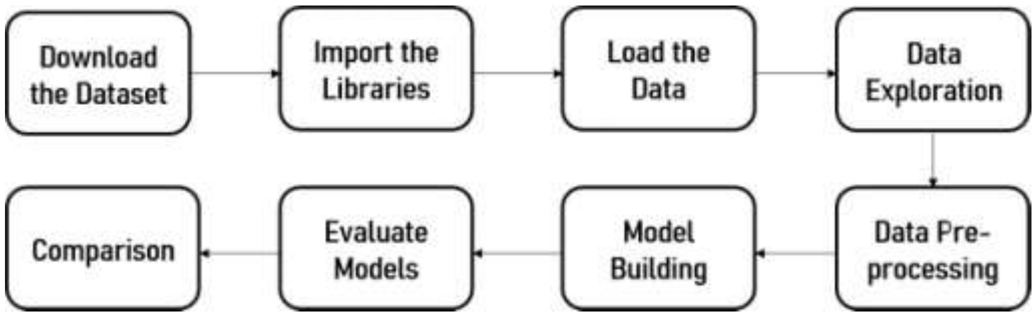

Fig 1: Steps for music generation using deep learning

1) Downloading the Dataset - A dataset is necessary to first train the model and then test its accuracy. We use the classical music MIDI dataset [10] which consists of piano compositions from 19 different artists such as Albeniz, Bach, Beethoven, Chopin, Haydn, Mozart, Schubert, Schumann, etc. From the above collection, we consider Mozart's composition to work on going further.

2) Important Libraries - We import libraries such as pandas and numpy along with a special library known as lilypond which allows us to print the MIDI sheet. We also import TensorFlow, an open-source library that allows the creation of large neural networks, Keras, which acts as an interface for the TensorFlow library, and music21 [11] to understand the musicology by computers.

3) Loading the Data - We load the songs in the Mozart's composition folder into a list and then parse them into a music21 stream. The next step is to separate the components of the MIDI files which are notes and chords. Notes are the building block of music and when a set of notes are played at the same time, a chord is formed.We save the extracted notes and chords in a list called the corpus.

4) Data Exploration - This section mainly involves exploring the data Corpus as well as the notes in the corpus and simplifying our corpus to build a working model. The input corpus is printed in the form of a conventional music sheet format with the help of music engraving software. The software we have made use of is called lilypond [12]. As the used input corpus is in the midi format this cannot be played directly on the used integrateddevelopment environment. Thus, we create an audio interface by converting the midi file into a ".wav" filetype. To simplify the large data corpus and prevent errors during model training, we have eliminated the notes occurring fewer than 100 times in the input corpus.

5) Data Preprocessing - Initially, we create a list containing all the unique notes and chords from the dataset. We then map all the unique notes and chords in our corpus to a specific number or id and create reverse-mapping of the key-value pairs to retrieve back the music at the time of prediction. Encoded corpus is divided into sequences of equal lengths of various features and their respective targets. These are one-hot encoded before being sent into the model for training. The input labels are resized and normalized before the model is developed. To train our model, we set aside 80 percent of our input data for the training purpose and the remaining 20 percent for testing the model.

6) Model Building - A Recurrent Neural Network (RNN) has the disadvantage of vanishing/exploding gradients when used on data requiring long-term memory [13]. The issue occurs due to the back-propagation of weights over several neural network layers. So, we are using a variant of RNN called Long Short Term Memory(LSTM) in our model architecture to learn the long-term dependencies of the music data. Due to their reliability,LSTM models have been used widely by researchers to generate music.

Unlike RNNs where we have only one tanh layer per module, LSTMs have four layers - three sigmoid and one tanh. The LSTM unit is shown in Figure 1. The hidden state of the previous module and the current input is concatenated and provided as input to the current LSTM module, which decides to what extent the previous cell state must be remembered. This flexibility to control the information that has to be retained, while discarding therest, makes LSTM immune to the vanishing gradient problem.

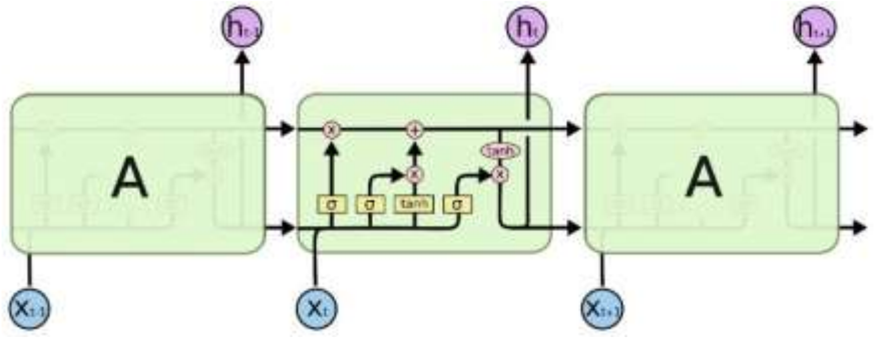

Fig 2 : The LSTM model

The hyper-parameters mentioned in the rest of this segment are our final choices after trying other values. The model performance evaluation based on others tested parameter values will be done under the Results section.

In our project, the LSTM must learn the input music’s characteristics to produce new music at the output while still retaining the original artist's style. Our model comprises two LSTM layers with 512 and 256 hidden states respectively. The first LSTM layer is followed by a Dropout layer with a rate of 0.1. This layer is used to prevent overfitting of the model to our dataset. Another Dropout is used after the first dense layer. The last layer is a fully connected dense layer with a softmax activation function. Softmax is used as we’re expecting the output to be probabilities for each class.

For compiling the model, we’re using the Adamax optimizer and the performance of our model is evaluated based on the loss metric called categorical cross-entropy. The loss curve is plotted to visualize the variation in loss over 100 epochs. A batch size of 512 is provided for every epoch.

7) Comparison - It can be difficult to discern two model-generated tunes unless one has a keen ear for music. So, we plotted the time graphs and frequency spectrums of the original corpus, and the two generated music files. Dissimilarity among the plots suggests the uniqueness of the output music relative to the input as well as other generated tunes from the same model.

Part 2: Implementation on Piano

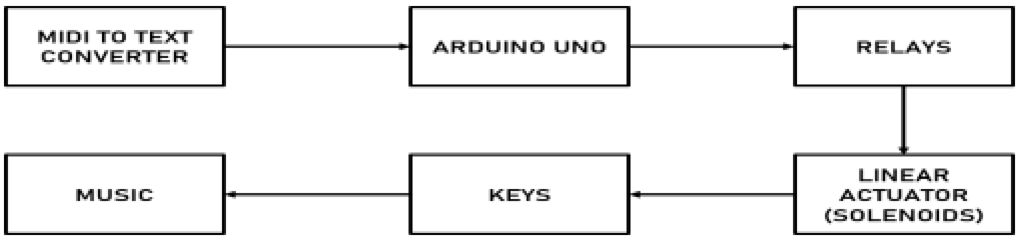

Fig 3: Block diagram for hardware

Automation of the entire keyboard or piano is an expensive task. Hence, for practical demonstration purposes, we first reduce the generated music to one octave, and then automate only 7 keys i.e., from note C4 to B4.

The music generated is in MIDI format. To work with arduino, the generated music should be converted into a text file. There are many online tools available for MIDI to text conversion. We have used [14] for this purpose. The text file has information about the note, whether it is pressed or not, and the velocity or duration ofpressing the note. Next, we extract the notes from the text file using Python code.

The project extracts each note and the numeric code associated with it. For example, the code for note C4 is 61, for D4 it is 63, for E4 it is 64, and so on. This code in binary form is fed to the arduino and based on this code, arduino generates the control signals. Arduino is a microcontroller and is used to control the relay board.

A relay is an electromechanical switch that closes or opens circuits depending on the control and works based on the principle of electromagnetic induction. Relays can be used in any one of the two configurations – The Normally Open (NO) configuration and the Normally Closed (NC) configuration. NO configuration is used when the circuit should always be open and close only when current is passed. NC configuration is used when the circuit should always be closed and open only when the current is passed. Since we want our keyboard key to be pressed only when that note is to be played, our circuit is always open and is closed only when that note is extracted. Hence all our 7 relays will operate in a Normally Open configuration. Each relay controls a solenoid.

The solenoid is a linear actuator or an electromagnetic actuator that converts electrical energy into linear motion. This also works based on electromagnetic induction. In our project, solenoids play the role of fingers pressing the keys to play the music. The current from relays causes the motion of the solenoids which isattached to the keys and the force from the motion causes the keys to be pressed.

IV. Results

In this section, the results of various hyper-parameter tunings observed while training the model are provided. When one hyper-parameter was being tested, we restored the original values for other parameters as discussed in the Model Building section. For instance, while experimenting with the Dropout layer's rate parameter, we used Adamax optimizer, 0.01 learning rate, and so on.

A) Dropout Rate - The dropout layer is used to prevent overfitting of the model. The parameter, "rate", defines the number of input units that will be dropped randomly at every step of the training. We tried three rates - 0.1,
0.2 and 0.3. The results obtained are shown in Table 1. It can be observed that the model performance when the rate is 0.1 is superior compared to the rest.

B) Optimizer - Optimizers are a significant part of the model training. They help improve the model by updatingthe weights with respect to the model's performance every epoch. The loss function is the metric that guides the optimizer towards the minima. Setting the dropout layer's rate to 0.1, three optimizers were used to analyze the model's performance. The results are shown in Table 2.

Dropout rate	Loss
0.1	0.1233
0.2	0.2956
0.3	0.2047

Table 1: Dropout rates versus loss

Optimizer	Loss
Adam	3.9099
Adamax	0.1233
RMSProp	0.1885

Table 2: Optimizer versus loss

C) Learning Rate - The learning rate defines how quickly the model gets to the minima of our loss function. Usually, a large step size is preferred at the beginning of training. Once the loss function gets close enough to the minima, the step size must be reduced to prevent overreaching and missing the minima. Setting the dropout layer's rate parameter to 0.1 and optimizer to Adamax, we trained our model using three different learning rates and the results are shown in Table 3.

D) Batch Size - The batch size is a parameter used when the model is fit to the data. The value that specifies the batch size is the number of X_train values that will be used in each epoch. We used three values and observed results as in Table 4.

Learning rate	Loss
0.01	0.1233
0.05	4.0363
0.1	3.9581

Table 3: Learning rates versus loss

Batch size	Loss
128	0.1230
256	0.1233
512	0.0307

Table 4: Batch sizes versus loss

V. Conclusion

In this paper, we present a deep learning autonomous approach to generating music using the LSTM modelon a dataset containing Mozart's music compositions. We also discuss how this generated music can be implemented on a keyboard using arduino and linear actuators to play the keys. The model learns the structure of the input music along with various patterns in it and predicts the music note or chord that must follow a given test sequence. We make use of a robust library called music21 that allows us to convert the midi dataset into a music21 stream for data preprocessing. We also explore how some of the hyper-parameters in our model architecture affect our model's performance.

Our current work majorly focuses on the music generation part and we only make use of monophonic data.In the future, this work could be extended to use polyphonic music and our hardware side can evolve from the implementation of only 7 keys on one octave to using all the keys on the keyboard to play the generated musicin its true form.

References

[1] Chen CC, Miikkulainen R. Creating melodies with evolving recurrent neural networks. InIJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222) 2001 Jul 15 (Vol. 3, pp. 2241-2246). IEEE.

[2] Liu I, Ramakrishnan B. Bach in 2014: Music composition with recurrent neural network. arXiv preprint arXiv:1412.3191. 2014 Dec 10.

[3] Eck D, Schmidhuber J. A first look at music composition using lstm recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale. 2002 Mar 15;103:48.

[4] Huang CZ, Vaswani A, Uszkoreit J, Shazeer N, Simon I, Hawthorne C, Dai AM, Hoffman MD, Dinculescu M, Eck D. Music transformer. arXiv preprint arXiv:1809.04281. 2018 Sep 12.

[5] Scimeca L, Ng C, Iida F. Gaussian process inference modelling of dynamic robot control for expressive piano playing. Plos one. 2020 Aug 14;15(8):e0237826.

[6] Malloch J, Sinclair S, Hollinger A, Wanderley MM. Input devices and music interaction. In Musical Robots and Interactive Multimodal Systems 2011 (pp. 67-83). Springer, Berlin, Heidelberg.

[7] Hayashi E. Automated piano: Techniques for accurate expression of piano playing. In Musical robots and interactive multimodal systems 2011 (pp. 143-163). Springer, Berlin, Heidelberg.

[8] S. Agarwal, V. Saxena, V. Singal and S. Aggarwal, "LSTM based Music Generation with Dataset Preprocessing and Reconstruction Techniques," 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018, pp. 455-462, doi: 10.1109/SSCI.2018.8628712.

[9] Hochreiter, Sepp, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies." (2001).