Music Generation using Generative Adversarial Network

Sunny Anand
Indian Institute of Science, Bangalore
Prajjwal Mishra
Indian Institute of Science, Bangalore

Abstract

Generating a piece of music by Generative Adversarial Network (GAN) that imitates a real work of musician has been a good topic of research since last few years. Many advancements in GAN implementations enabled researchers with a good tool to generate some decent music. The model we propose here is a continuation of research work going on in this field. We worked with some music samples in form of midi file format and used the information inherent in it to produce a new music sample. We conclude that using all the information in message bytes of a midi file improves the quality of music produced, evaluate the music on some specified metrics, and let the listener judge the quality by listening to it.

1. Introduction

Music is an art form which expresses emotions through harmonic frequencies. Many amazing composers in past has produced some good pieces that is both creative and deliberate. But can we generate some piece using machines that can stands against that great works for at least some extent?

In this work we're trying to find out whether we can build such generative model that can really understand the notion of harmony and melody inherent in a piece of music and based on that, produce a good music to listen.

2. MIDI Data[1]

Midi files are structured as a series of concurrent tracks, each containing a list of meta messages and messages. We are considering only note messages as our input data and training our generative model with that.

Each note message is a combination of three bytes. The first byte is called **Status Byte**. A **Status Byte** has MSB set to 1. Last 4 bits of **Status Byte** identify the midi channels (0 to 15). Also, the fourth most significant bit is set if note is on and reset if note is off. A **Status Byte** is followed by 2 **Data Bytes**. Both **Data Bytes** has MSB set to 0. Last 7 bits of first **Data Byte** identifies the pitch value of the note (ranging 0 to 127). Also, the last 7 bits of second data byte contains information about velocity value (ranging 0 to 127).

3. Background and Previous Works

All the previous works on music generation that includes messages in the midi files only extracted the features of data bytes (which contains the information about notes) and worked with them. The paper we're referring here is **C-RNN-GAN**[2] in which the author took notes of a given midi message and operated on the notes but we operated on notes , velocity and status byte of a midi message .

4. Experimental Setup

Dataset: Training data is collected from the Github[2] link in the form of midi file , are well known work of Artists . The data used contains 36 midi files composed by different composers .

Hardware : We Used **Google Colaboratory** for the purpose of training and testing the model equipped with gpu backend .

5. Approach

A general adversarial network(GAN) consists of 2 neural networks Discriminator(D) and Generator(G) is used in our Work.

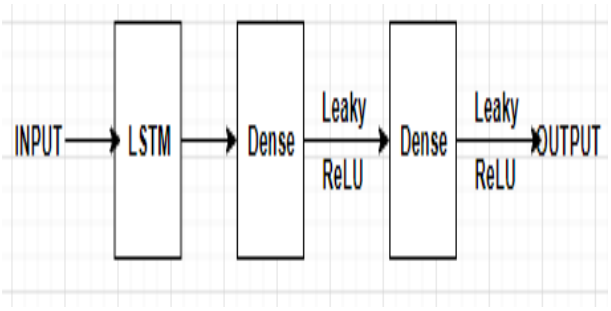
- A neural network called "Generator " which generates new data points from some random uniform distribution. The goal is to produce the similar type of fake results from inputs.
- while another neural network called "Discriminator" which identifies the fake data produced by Generator

from the real data.

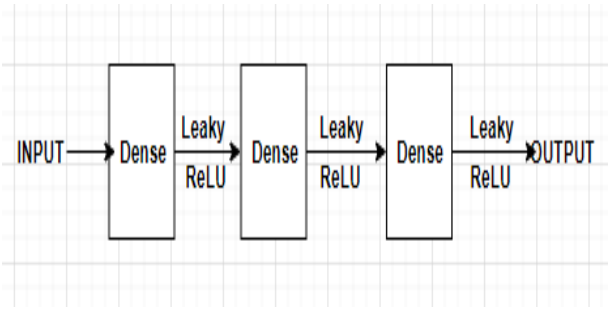
Discriminator : Our discriminator model is consist of 1 LSTM layer with 512 nodes , 1 Bidirectional LSTM with 512 nodes and 2 dense layers one with 512 nodes and another with 256 nodes with Activation function "Leaky Relu"

Generator : Our generator model is consist of 3 Dense layers each with 256 nodes with Batch Normalization Activtion Function "Leaky ReLu" and "tanh" .

- **Discriminator Model**



- **Generator Model**



At Discriminator D :

$$Dloss_{real} = \log(D(x))$$

$$Dloss_{fake} = \log(1 - D(G(z)))$$

$$Dloss = Dloss_{real} + Dloss_{fake}$$

$$TotalCost = \frac{1}{m}\sum_{i=1}^{m} \log(D(x)) + \log(1 - D(G(z)))$$

At Generator G :

$$Gloss = -\log(D(G(z))$$

$$TotalCost = \frac{1}{m}\sum_{i=1}^{m} -\log(D(G(z^i))$$

$$min_G max_D V(D,G) =$$

$$E_{x\ p_{data}(x)}[\log D(x)] + E_{z\ p_z(z)}[\log(1 - D(G(z))]$$

6. Evaluation

Evaluation of music is a tough task , we have a question that how "good" our model is at generating "good" music , for this purpose we used a combination of human and Criteria based evaluation . We consider a model better if average metric value of generated sample is close to original sample and peoples asked to evaluate music reported it good .

- **Qualified Note Rate:** ratio of number of notes greater than 32^{th} note (termed as Qualified Note) , *low QN implies overly fragmented music*[3]
- **Human Based Evaluation :** We Asked some of our colleagues to listen to the music generated and tell us whether it is good or bad .

7. Results and Observations

7.1. Qulified Note Rate :

- **QN Rate for input sample :** 0.933
- **QN Rate for generated sample :** 0.792

7.2. Human Evalution :

We gathered 5 of our colleagues and let them listen and judge the quality of music and give a rating on the scale of 1-5. 2 out of 5 rated it 2.5 , one rated it 2 , and other 2 rated it 3 .

References

1. MIDI Tutorial : *http://www.music-software-development.com/midi-tutorial.html*
2. Implementation of C-RNN-GAN *https://github.com/olofmogren/c-rnn-gan*
3. Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation : *https://arxiv.org/abs/1804.09399*
4. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment : *https://arxiv.org/pdf/1709.06298.pdf*