Music Engendering Algorithm With Diverse Characteristic By Adjusting The Parameters Of Deep Convolutional Generative Adversarial Networks

Aitik Dandapat[1], Dr. Aishwarya Dandpat[2]

[1]Undergraduate Student, Veer Surendra Sai University of Technology, Burla
[2]DNB Resident, Ramkrishna Care Hospital

Abstract - *With the widespread development of deep learning, automatic composition has become a highly topical topic occupying the minds of music computer scientists. The paper proposes advanced arithmetic for music engenderation using Generative Adversarial Networks (GANs). The music is divided into tracks and the note segment of the tracks is expressed as a piano roll by a trained GAN model whose generator and discriminator play a continuous zero-sum game to produce high musical integrity.*

In most cases, although GAN excels in image generation, the model adopts a cross-channel deep convolutional network structure in accordance with the properties of the music data in this article, producing music more closely matched to human hearing and aesthetics.

Key Words**: Generative Adversarial Network, DGCAN, Convolutional layer, MusPy, Pianoroll**

1. INTRODUCTION

Music Engenderation composes music on the computer using machine learning methods. With the increase in AI data and research, researchers have started exploring machine learning in music creation.

Deep learning is a branch of machine learning methods that can recognize patterns and make decisions without explicit programming. It shows promising results in several areas such as natural language processing for texts, computer vision for images, and speech recognition for speech. In addition, deep learning techniques in artificial music engenderation have successfully generated human-like compositions. However, most research has focused on musical composition and neglected the aspect of expressive musical performance. Therefore, musical information stored on MIDI (Musical Instrument Digital Interface) tracks, such as speed, proved useless during practice. This makes the music produced rather mechanical and boring.

This document concerns the design of a music production system that could produce music with built-in speed, also known as musical dynamics. To do this, the training data must encode speed as well as altitude and time information. The piano roll (a picture-like representation of data) was used to encode information, with one axis representing tempo and the other axis representing pitch. Each pixel intensity in the range of 0–128 was represented by the note's velocity. DCGAN was chosen as the deep learning architecture used in this article. It can capture and learn the data distribution from a data set and generate a sample from the same distribution. Finally, the model generates sample snippets: some are synthesized with musical dynamics and some are not. These extracts were mixed with normal music for analysis and a user study was conducted.

2. RELATED WORK

Algorithmic composition is a subject of work that dates back to 1959. However, the recent developments of Deep Neural Networks (DNN), which have proven astonishing results in learning from big datasets, allowed this topic of music generation to be further developed. Over the past couple of years, tons of proposed models addressing music generation have been published, all of them on deep learning algorithms.

2.1 Recurrent Neural Network

The most popular architecture in music creation systems is the recurrent neural network (RNN). It is a feedforward neural network, meaning that the output from the hidden layers is transmitted back to it and the layer below it. As a result, it has the capacity to record information and pick up knowledge from sequence data. However, because the weight matrix is updated continuously during backpropagation, RNN experiences the vanishing and exploding gradient problem.

Long Short-Term Memory (LSTM), an RNN variant, uses a variety of gates to overcome the issue in a unique RNN. These gates include input, output, forget, and cell state gates. Information can be sent through cell states as a conduit. An applicant is chosen by the input gates. The input gates pick a candidate from the inputs and change of any previously available pertinent data. The forget gates, on the other hand, purge the cell states of unimportant data. The information that is sent to the next concealed state is finally decided by the output gates.

A Google group called Magenta utilized LSTM to create the expressive music creation system Performance RNN. It takes advantage of the MIDI tracks' velocity, pitch, and note durations. It was noted that the music had dynamism and phrasing. It could not, however, show long-term structure.

2.2 Generative Adversarial Network (GAN)

Impressive results were obtained when the Generative Adversarial Network (GAN) was used to create music. It comprises two network models that are engaged in a zero-sum game: a generator and a discriminator. The goal of the generator is to deceive the discriminator by converting a random noise input into a sample that matches the distribution of the real samples. The discriminator's goal is to separate a genuine sample from a produced sample, nevertheless. Following the Nash equilibrium of both network models, the discriminator is eliminated.

The Music and AI Lab in Taiwan created MuseGAN, a system that can create multi-track music without the need for human input. It converted MIDI recordings' note length and pitch into a piano-roll format. To create the multi-track piano roll that served as the training data, various musical instruments were encoded into several piano rolls and piled together.

2.3 DCGAN

The Deep Convolutional GAN (DCGAN) model is no more than a GAN whose discriminator and generator networks comprehend some convolutional layers, such as CNNs do. The work of Radford, Metz, and Chintala led to the adoption of some already proven modifications to the CNN standard architecture. The major components of this model are Strided Convolutions, Fractional Strided Convolutions, Eliminating Fully-Connected Layers, Batch Normalization, Generator and Discriminator.

The convolution has a favorable effect on extracting picture features and employs convolution in place of the fully connected layer. Step size convolution is used in place of the upsampling layer. More importantly, we use the full-channel lateral convolution kernel, which enables the model to learn Be more focused on musicality and converge more quickly because, since the piano roller blinds are a form of music, we don't need to worry too much about the longitudinal convolution.

3. PROPOSED MODELS

All of the suggested models will be discussed in this paper. These adhere to the DCGAN's nature. Despite slight modifications, the convolutional approach and the suggested changes by Radford, Metz, and Chintala are adopted here.

All of the models provided in this paper share the core of the method. The focus now is on each network's own structure because the dynamics between the discriminator and generator networks were already discussed in the previous section.

To construct the random vector z, the method begins by selecting 100 samples from a normal distribution N (0, 1). This vector will be input into the generator network and pass through a layer of linear operations, with the bias initialized to zero and the weight matrix initialized at random from a normal distribution N (0, 0.2). The output of this linear layer is then batch-normalized, 3-dimensionally reshaped, and subjected to the ReLU non-linearity. Convolutional layers are used in the subsequent phases; their parameters will be determined later, once they vary depending on each proposed model. With the exception of the last one where the Tanh nonlinearity is used, each convolutional layer's input is batch-normalized, and its output is subject to the ReLU nonlinearity.

An input for the discriminator network is a three-dimensional form of created or actual data. Convolutional layers are used in the subsequent phases; their parameters will be determined later, once they vary depending on each proposed model. The LReLU with a leak of 0.2 is still applied as a non-linearity after each convolutional layer, and all convolutional layer inputs—with the exception of the first—are batch-normalized. Following the convolutional processes, the three-dimensional output shape is flattened and subjected to a linear layer, the parameters of which are initialized as the linear layer used in the generator network, to create a single output unit. A sigmoid nonlinearity is added to this final unit.

Three models are proposed with different approaches to the convolutional steps.

3.1 Original DCGAN - Model 1

The DCGAN model from the beginning is fairly similar to Model 1. The depth of the networks has to be modified to only 3 convolutional layers each due to the varying dimensions of the data. Although the filters still have relatively small spatial extents and the strides are also of the same order of magnitude, both the filter's and the stride's dimensioning have changed.

Table 3.1 provides details on the corresponding convolutional layer parameters.

Table -1: Model 1 convolutional layer's

Network	Layer	Input	Stride	Output
Generator	Conv 1	17×23×256	5×5	85×115×128
	Conv 2	85×115×128	4×3	340×345×64

Network	Layer	Input	Stride	Output
	Conv 3	340×345×64	2×1	680×345×32
Discriminator	Conv 1	680×345×32	2×1	340×345×64
	Conv 2	340×345×64	4×3	85×115×128
	Conv 3	85×115×128	5×5	17×23×256

3.2 Strided Width - Model 2

In this model, it is suggested to move along the breadth, which implies that the receptive field will only take into account the entirety of the action taking place at each temporal interval. This makes sure that the analysis of the frequency component, or the notes being played, is entirely the responsibility of the first convolutional layer. The temporal link between each time period will therefore be controlled by the subsequent convolutional layers.

Figure 1 shows the architecture of the Model 2 networks, while Table 2 lists the parameters for the corresponding convolutional layers.

Table -2: Model 2 convolutional layer's

Network	Layer	Input	Stride	Output
Generator	Conv 1	1×23×256	1×5	1×115×128
	Conv 2	1×115×128	1×3	1×345×64
	Conv 3	1×345×64	680×1	680×345×32
Discriminator	Conv 1	680×345×32	680×1	1×345×64
	Conv 2	1×345×64	1×3	1×115×128
	Conv 3	1×115×128	1×5	1×23×256

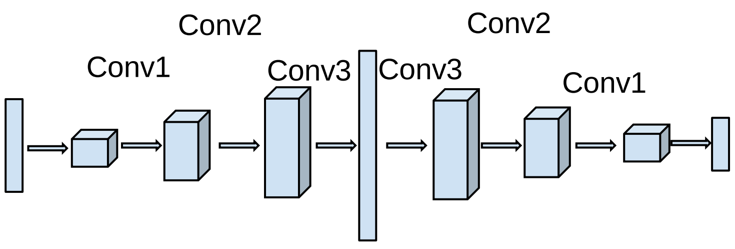

Fig -1 Generator & Discriminator Network Architecture for model 2

3.3 Strided Height - Model 3

In a way, Model 3 is the antithesis of Model 2. This model suggests "stumbling along the height," which implies that the receptive field will evaluate the whole time series while covering one frequency interval at a time. This makes sure that the analysis of the temporal component, or the determination of which time-steps a particular note is being played, is entirely the responsibility of the first convolutional layer. Therefore, the relationship between various notes will be controlled by the following convolutional layers.

Figure 2 shows the architecture of the Model 3 networks, while Table 3 details the specifications of the corresponding convolutional layers.

Table -3: Model 3 convolutional layer's

Network	Layer	Input	Stride	Output
Generator	Conv 1	17×1×256	5×1	85×1×128
	Conv 2	85×1×128	3×1	340×1×64
	Conv 3	340×1×64	1×345	680×345×32
Discriminator	Conv 1	680×345×32	1×345	340×1×64
	Conv 2	340×1×64	3×1	85×1×128
	Conv 3	85×1×128	5×1	17×1×256

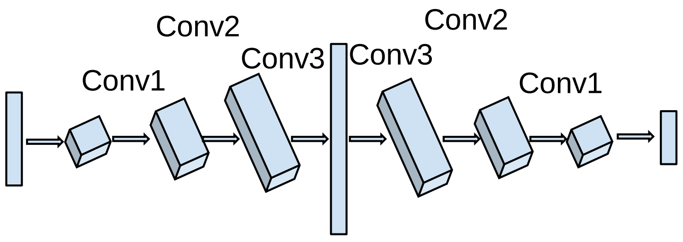

Fig -2: Generator & Discriminator Network Architecture for mod 3

5. IMPLEMENTATION

The model explained in the previous section was implemented, and the results were captured. All the steps involved and points taken into consideration while performing the experiment are explained under the following subtopics. Results were obtained after the experiment was evaluated to determine the precession of the study.

5.1 Dataset

There are 100 MIDI tracks and 103 WAV files in Cymatics' Oracle Hip Hop Sample Pack. We solely utilized MIDI recordings in this work since they carry musical data in a symbolic manner. We used MIDI tracks with durations of 8 or 16 seconds and records with minor tonality, as the bulk of the songs fell into this category, in order to simplify the training data. The notes are composed of a beat, a pitch, a velocity, and a duration.

The open-source Python module MusPy, which generates symbolic music, was used to extract the MIDI data, including pitch, velocity, and note duration. With the use of MusPy tools, this data was subsequently encoded into a piano-roll data representation.

5.2 Data Pre - Processing

The MIDI file is loaded into the program using MusPy tools for data pre-processing, where it is parsed into a music object with a temporal precision of 12 timesteps per beat. Since the music object is inaudible to the human ear and can only be transposed to the key of C major, it was restricted by clipping its velocity to at least 40. We chose to extract just up to 4 octaves of pitches (48 notes) because the musical piece will not contain all 128 pitches. The training data's output dimension was (N, 48, 48, 1), where N is the number of training samples and (48, 48, 1) is the size of a single piano-roll.

5.4 Generator and Discriminator

The generator transforms (100,1) random vectors into M*N*C as inputs. The letters M and N stand for the height and breadth, respectively, and the letters C stand for the number of channels. Then, it goes via 3 Conv2DTranspose layers with 128, 64, and 1 filters that, respectively, activate Tanh, ReLU, and ReLU. The generator's output dimension will be the size of a piano roll with the coordinates (48, 48, 1).

It was developed for the discriminator in the opposite order from the generator. It receives the piano-roll as input and runs it through three Conv2D layers with the LeakyReLU activation function. The output value was then predicted using a Sigmoid activation function after it was flattened and passed through three dense layers.

5.5 GAN Training

The training of the generator and discriminator uses the Adam optimizer. The GAN's two sides can both overwhelm the other. The generator will have trouble reading the gradient if the discriminator model is overtrained since it will output values that are so close to 0 or 1. Otherwise, it will repeatedly take advantage of discriminator flaws that result in false negatives.

We discovered during the experiment that we can run as many iterations as necessary. After all, the goal of the music creation algorithm is to produce better-sounding models rather than massive ones.

GAN model training takes quite a bit, therefore, if parallel GPU training is possible, we can train GAN models more quickly than we could otherwise. The Nvidia GeForce RTX 2080 graphics processing unit was utilized.

5.5 Data Post - Processing

The resulting piano roll was padded to a size of (128, 48, 1) to go back to the original piano roll, same as before data preprocessing. MusPy library methods were used to transform the piano roll back into a MusPy music object, which was then written into a MIDI track in order to create MIDI tracks from a post-processed piano roll.

6. EVALUATION

A very subjective topic is music. Right now, hearing is necessary in order to evaluate the musical quality. We thus employ two sets of programs to assess the outcomes of our experiments. We conducted a poll with two groups: experts in the music industry and others who just sometimes listen to music. We blended in some generated music by our model and a variety of song genres that we randomly listen to and untrained music. A few simple questions were posed, along with a rating system for the music on the basis of its softness. These replies show that the method for creating music is capable of producing melodic and harmonic progressions that are engaging.

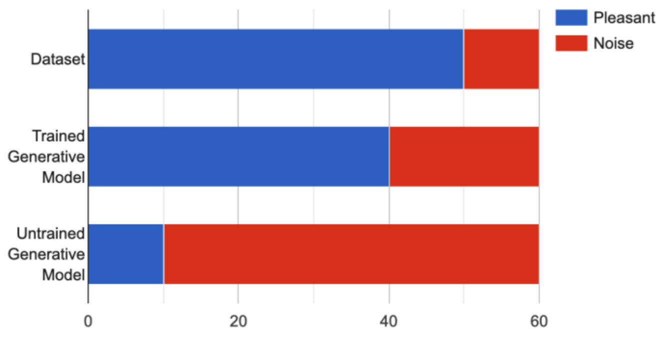

Chart -1: User Study

7. CONCLUSION

In this study, we present a generation model for the GAN framework to generate note sequences. We employ a deep convolutional neural network and tune it to the properties of musical notes; this optimization approach allows the convolution network to concentrate on picking up musical information and conducting tests more quickly. In order to hasten the training of discriminators, we also get some understanding of common sense at the same time. Experimental results and arbitrary user assessments demonstrate the suggested model's ability to produce musically inspired sequences. The model is theoretically developed and possesses optimal qualities, despite the experimental results being below the level of human performance. A future study on the use of GAN for music production may concentrate on producing music of any duration or collecting a bigger training data set.

REFERENCES

[1] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, Midinet: A convolutional generative adversarial network for symbolic-domain music generation, 2017. eprint: arXiv:1703.10847.

[2] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017. eprint: arXiv: 1709.06298.

[3] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, Wavenet: A generative model for raw audio, 2016. eprint: arXiv: 1609.03499.

[4] V. Kalingeri and S. Grandhe, Music generation with deep learning, 2016. eprint: arXiv : 1612 . 04928.

[5] A. Nayebi and M. Vitelli, "Gruv: Algorithmic music generation using recurrent neural networks", Course CS224D: Deep Learning for Natural Language Processing (Stanford), 2015.

[6] A. Eigenfeldt and P. Pasquier, "Realtime generation of harmonic progressions using controlled markov selection", in Proceedings of ICCC-X-Computational Creativity Conference, 2010, pp. 16– 25

[7] "Cymatics.fm - The #1 Site For Serum Presets, Samplepacks & More!" https://cymatics.fm/ (accessed 4 Oct, 2020).