Generating Music by using Deep Learning

Furkan Karadeli Nermin Nur Aydoğan Erhan Kabaoğlu

Abstract

The study of music generation first stepped forward in with the WaveNet model introduced by van den Oord et al. (2016) in the deep learning field and has been drawing attention recently since the construction of the Magenta project. Consequently, researchers have increased their researches on the topic to explore the role of machine learning as a tool in the productive process. With this aim, several methods are discussed to produce audio samples. In this study, we propose Transformer-based audio generator model. We use piano as our instrument and for this purpose we use Maestro Dataset.

1. Introduction

Generating music is challenging for artists. Finding right notes and combined them meaningfully is very hard. When artists make composition sometimes they stuck and they waiting inspiration. Musician inspired by this generated music and these musics can be based of different compositions. To solve this problem, we tried generate piano music with deep learning using MIDI recordings from ten years of International Piano-e-Competition. Generating piano music is challenging because piano has 88 different notes and each timestamp has more than 1 notes. Since whether result is good or not good depends on person, testing our model is challenging too. In recent years , there have been works how to generate music with using Recurrent Neural Networks(RNN), Transformers and Generative Adversarial Networks(GAN). In our work we firstly tried Transformers and we tried to combine Transformers and GANs but there is no example about this combine, so it is so hard to applied. Generator takes random noise vector and gives music sequences. Discriminator takes these music sequences and classify them. After build the models, we train our model with adversarial manner.

2. Related Work

Today, many architectures have been tried with the developments in the field of deep learning for music generation.

2.1. RNN Based Models

RNN architecture has been used in NLP problems for a long time, but the RNN model has some weaknesses when capturing long term dependencies. Architectures such as Lstm and Gru were developed to overcome this problem. Various Lstm architectures used for symbolic music generation [1, 2, 3].

2.2. CNN Based Models

RNN based models takes one token at a time so this cause major performance decreasing. Therefore, researchers started to use CNN based models in the NLP tasks to preserve parallelisation. Almost every music has very long sequences so this cause critical performance issues. There are different type of cnn based architecture used in symbolic music generation task. MidiNet is one of those architecture [4]. It uses Generative Adversarial Network architecture to generate symbolic music. Generative Adversarial Network consist of two parts which are generator network and discriminator network. As typical in GANs, the input of G is a vector of random noises (z E R) l , whereas the output of G is an h-by-w matrix Xb = G(z) that "appears" to be real to D. GANs learn G and D by solving:

MidiNet reached state of art performance with GAN's. Variational Autoencoder which is one of the other generative models, was also used frequently for symbolic music generation [5]. Variational autoencoder consist of two parts which are encoder and decoder. The encoder, denoted by q(z—x), encodes a data sample x to a latent (hidden) representation z: z ˜ q(z—x). The decoder, denoted by p(x—z), decodes the latent representation back to the probability distribution of the data (in data space): xˆ ˜ p(x—z). The VAE regularizes the encoder by imposing a prior over the latent distribution p(z) where z ˜ N (0, I). The loss function of the VAE is the expected log likelihood with a regularizer:

2.3. Transformer Based Models

As we mentioned above, RNN based models do not perform well as they takes one token at each timestamp in the training phase. On the other hand, CNNs only apply the Convolutional operation to certain areas. Therefore, they can't track down long term dependencies in the input sequences. Transformers offer solutions to these two problems

at the same time. This architecture, first presented in the article published by Google Brain team, performed much better than other models in many computer vision and NLP tasks today [6]. Transformers consist of mainly two parts which are encoder block and decoder block. These blocks contains multi-head attention and feed forward blocks. The main difference compared to RNN based models is that it takes input data at once and can find long-term connections between input and output using the self-attention mechanism. There are different studies to generate symbolic music using transformer architecture. MusicTransfomer is one of them published by Google Brain team using a single transformer [7]. Also there are some hybrid architectures like Transformer with GAN's for symbolic music generation [8].

3. Approach

In our work, we tried 2 different models. First one is Transformer, second one is combined Transformer and GANs.

3.1. Transformer

In Piano Music Transformer we inspired this code[9]. Code designed to train Neural Network on piano MIDI data to generate samples in Python. MIDI files are encoded into "Event Sequences" a dense array of piano music encoded as numerical tokens. In training phase we used Adam optimizer as a optimizer and we used Cross-Entropy loss function but after generate samples we used human evaluation. Our epoch number is 5 because training takes so many time. Our model has Decoder Layer and we applied layer normalization to this layer. In decoder layer we have another layer which name is PositionwiseFeedForward. This layer has 2 linear layer and one droput layer. Droput value is 0.1. As an activation function we used ReLU function in this layer.

3.1.1. Relative Attention

Musical notes are building structures of piano musical pieces. These building structures with repetitions are called performance events in transformers modeling. Likewise a word in a sentence, these performance events are considered as a token to a musical piece. Thus, these tokens reveal a relationship between each other. Self-attention enables us to learn the relationship between these tokens. Relative attention is self-attention with relative position representation. The main focus of a Transformer is built on relative attention.

Each component of the input sequence is summed by weights correspondingly to their positions to obtain the output sequence. This weighted sum is called 'attending'. This weighted sum has three key components: query, key, and tensor values which are represented as Q, K, V. The following formula shows how attention is calculated for each input sequence.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V$$

Figure 1. Attention Formula

The distance between two positions is significantly important for relative attention. The distance factor strengthens relative attention's efficiency. This is due to position representation that derives information of the pattern's structure. These pattern structures give an idea about motifs, which are the core of piano music pieces. Input sequences can be represented as a fully connected graph, and from fully connected graphs, we can construct their tensor representation.

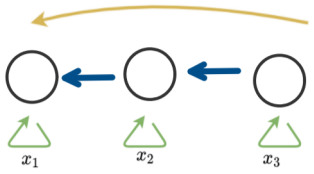

Figure 2. Tokens as Fully Connected Graph

As an overview, input sequence is divided into blocks. These blocks are distinctly considered, they only attend themselves.

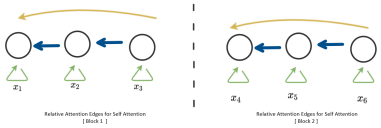

Figure 3. Tokens into Blocks

From fully connected graph, edges and distances are represented through tensors. The information of distances of edges are called embeddings. Embedings encode the relative position between position of current token from query tensor, and position of current token from key tensor. This process happens through some stages. Firstly, to get relative position representations, the query tensor and embeddings are multiplied. Then, some masking, padding and reshaping operations are done. The stages are explained in the given figure below.

Figure 4. Embedding to Tensors

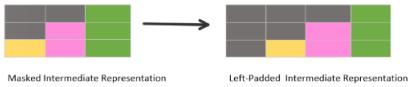

Figure 5. Intermediate Representation

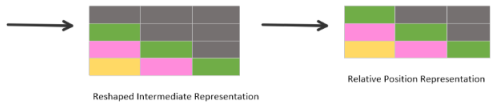

Figure 6. Intermediate Representation cont.

Lastly, a skewing procedure is applied to obtain requested tensor. This skewing includes slicing as well. These stages are progressed through as given figure below.

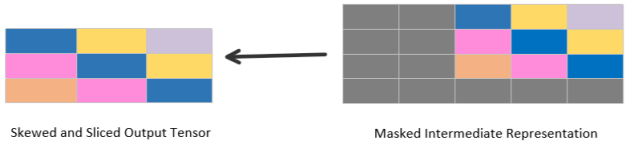

Figure 7. Skewing Step

3.2. Combine Transformer and GAN

There are many researches for Music Generation using Generative Adversariel Neural Networks. Gans was first used in the field of computer vision. Many different models have been tried so far. These models generally used a convolutional neural network as a backbone. When we examined the results, we saw that GANs are a very powerful generative model. But in our problem, we thought that using convolutinal based gans was not enough to catch long dependencies. That's why we tried to combine Transformers and Gans, which have been very popular lately.

Gans consist of two different networks as discriminator and generator. After the generator receives the random noise vector, it passes through certain layers and produces a sample. After that, the discriminator tries to distinguish whether this sample is real or fake after taking this sample as input and passing through certain layers. In this way, the generator tries to learn the distribution of the given data thanks to the feedback it receives from the discriminator. In our model, we tried to combine the transformer architecture we mentioned above with the gan model. In other words, we tried to do adverserial training using transformers.

For the generator, we had to first remove the embedding layer. because embedding layer takes discrete values so it is not suitable for random noise vector. We replaced the embedding layer with the traditional feed forward network. This feed forward network takes a noise vector DxZ. D corresponds to the shape of the embedding vector. Z corresponds to the size of the random noise vector. We chose to try such a method so as not to disturb the structure of the transformer. But removing the embedding layer caused the given input to be represented poorly. The rest of our model has the same architecture as the transformer model used in the music transformer model we mentioned above. So the network gives an output in the form of S x N. S corresponds to a fixed value of music tokens, and N corresponds to the number of features we get after preprocessing the music data.

Discriminator takes music sequence with shape A. A corresponds to the argmax of the features we obtained from generator. This stage is the same as the music transformer architecture already mentioned above. This transformer gives an output in the form of SxN. We applied global average pooling to this output. In this way, the output we obtained has become 1xN. Finally, we used the traditional feed forward network for classification. This feed forward network returns a single value, taking the output with shape 1xN.

After we created our generator and discriminator models, we started doing adversarial training.

4. Experimental Result

4.1. Dataset

The dataset contains over 2,000 performances played by professional pianists which is available on The Maestro Dataset. For train our model we used piano MIDI files. Dataset is splitted into Train(%70), Validation(%15) and

test(%15).As we showed in Relative Works, we decide use Transformer Based Models. For try this experimen used a small dataset which is 100 MIDI files.

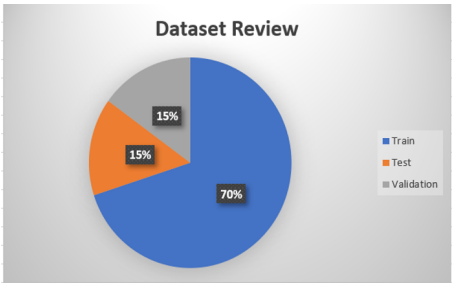

Figure 8. Dataset Review

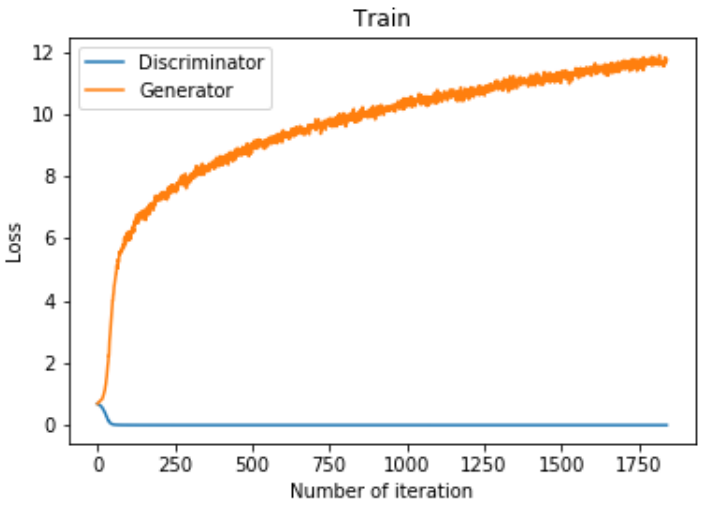

Figure 10. Trasnformer-Gan Loss

4.2. Hyperparameters

In this work we tried different parameters. One of them is batch size. We tried 8 and 16. There is no important different between them. Also we tried droput values such as 0.1, 0.2 and 0.01. Best droput value is 0.1. Number of heads are 8 for multi-head attention. Noise vector size is 50.

4.3. Results

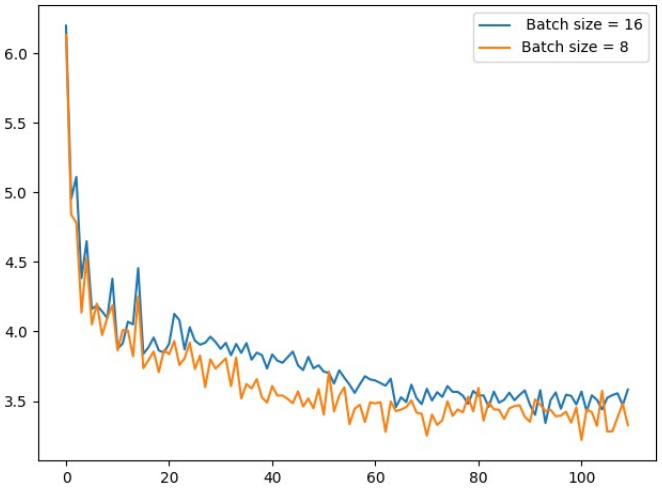

Figure 9. Transformer Loss

You can see loss plot for batch size 16 and batch size 8 using transformer.

We trained our transformer gan model for 10 epochs. and the discriminator outperfom the discriminator. Because the discriminator's job is much easier than the generator. So this caused mode collapse in generator. We also trained our model from a small number of datasets due to time constraints. For this reason, we could not make our generator learn.

5. Conclusions

We evaluated our model through human evaluation. Out of 10 people, 5 people considered the result audio as good, while 4 people considered average and 1 person evaluated as bad. Participants agreed that the model can be improved to result better outcomes. Additionally, participants shared their ideas about audios. Majority of the participants, 6 of them, found the music generated by model inspiring.

Figure 11. Result Review

However, there does not exist a conventional evaluation on

music generation since music is subjective and does not hold objective properties. Several methods are used in evaluation. In this study, we also benefit from cross entropy loss to adjust hyperparameters of the model to achieve effective results on human evaluation.

Future work of music generation is convenient to be improved. Sufficient amount of data is needed to work effectively on models. We believe present data can be increased to expand the researches on the music generation topic. This data can also further categorized into emotions since emotion plays a significant role in real-human-life music generation. Apart from that, as a future work, generating music with specific style according to the users' requirements can be considered. Such style examples can be listed as Classical Piano,Jazz Piano, Musical Theater Piano, Pop Rock Piano To achieve this, real-time interaction with users can be utilized in the process.

References

[1]Deep Learning for Music, Allen Huang and Raymond Wu
[2] Modelling High-Dimensional Sequences with LSTM-RTRBM: Application to Polyphonic Music Generation, Qi Lyu, Zhiyong Wu, Jun Zhu, Helen Meng
[3] LSTM Based Music Generation System, Sanidhya Mangal,Rahul Modak, Poorva Joshi
[4] WAVENET: A Generative Model for Raw Audio, Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu
[5]Semi-recurrent CNN-Based VAE-GAN for Sequential Data Generation, Mohammad Akbari and Jie Liang
[7] Music Transformer: Generating Music with Long-Term Structure, Cheng-Zhi Anna Huang, Ashish Vaswani Jakob Uszkoreit Noam Shazeer Ian Simon Curtis Hawthorne Andrew M. Dai Matthew D. Hoffman Monica Dinculescu Douglas Eck
[8] Symbolic Music Generation with Transformer-GANs, Aashiq Muhamed, Liang Li ,Xingjian Shi, Rahul Suresh, Alexander J. Smola
[9] https://github.com/chathasphere/pno-ai