A.I. - Final Project Report

Project Name: Music GAN

組員：0816032李懿麒 0816034蔡家倫 0816035陳威達 0816102陳品戎

Abstract

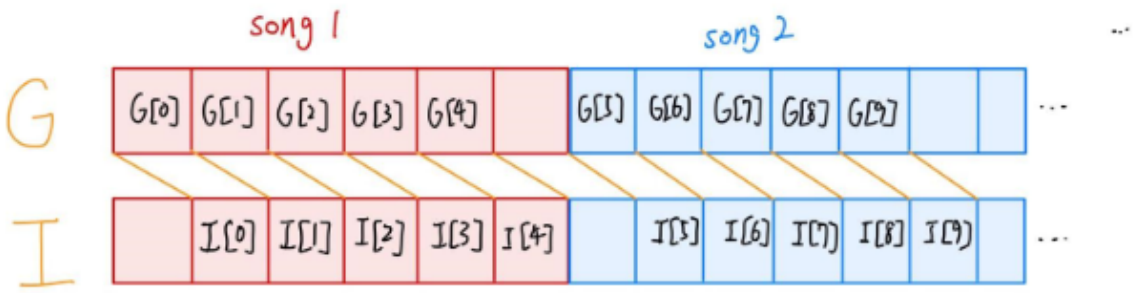
We try to use GAN (Generative Adversarial Network) to generate music of J. S. Bach Chorale-like genre. Time-related data like music may need models with memorization features like RNN, but CNNs recently show similar performances with higher efficiency. Thus, we chose DCGAN as the base of our neural network. As for the training data, considering noise on audio file formats like `.mp3` or `.avi`, we chose `midi` files considering the professors' suggestion and the fact that it is a representation of pure sheet music and noise-free.

Data Processing

We use the Python's `MusPy` library to parse music and change it to Pytorch's Dataset format, and used the preorganized `JSBChoralesDataset` to easily load our training data.

We split all songs in the dataset to 11418 bars (小節) and formed 2 lists G and I out of the sequential data, with the two arrays as sources of the generator's and discriminator's input respectively. Each bar's data is a 128 by 16 0-1 matrix, with the two axises representing the pitch and time (with a sixteenth note as a unit) of a note, known as a piano roll. To make it simple, we exclude the velocity(volume) factors and use a binary entry to indicate whether a note is played at the time.

We wish the generated music inherits correlations on neighboring bars, thus we process data in a way shown in the diagram below. For each time step, the section from G is one bar ahead of the section from I, thus we use G as a condition of the generator and try to make the fake data match the next bar, which is I.



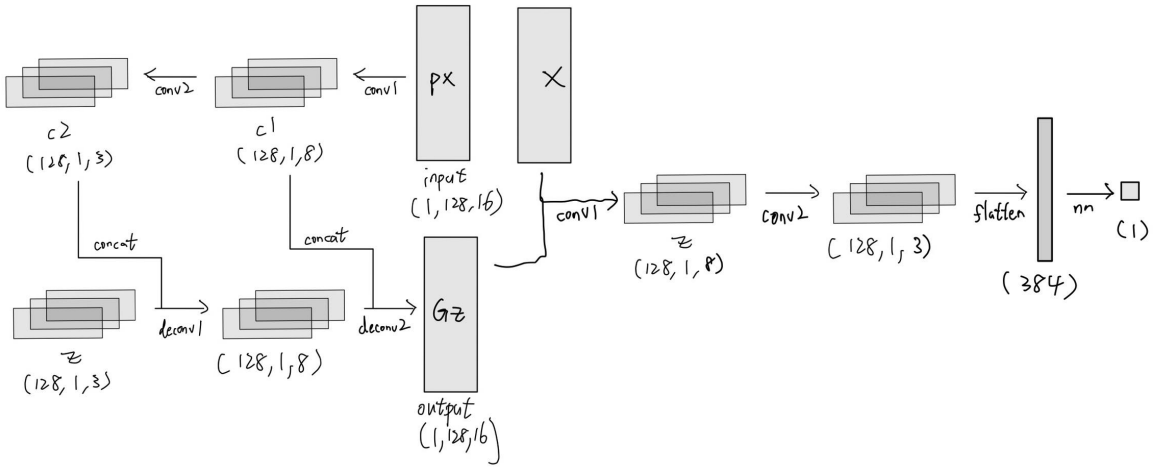
G is one bar ahead of I, incomplete bars will be deposited

After data preprocessing, we end up with data of 10240 bars on both arrays, wrapped them into our custom dataset and it as a .pt file to train on Pytorch models later on.

Model & Training

As like most GAN structures, we used a generator to make data from noise and use the discriminator to classify the true and fake piano rolls. We took ideas from MidiNet proposed by Academia Sinica, since our ideal input and output format is identical to this model. We read the paper and implemented the model with Pytorch Lightning. The discriminator is nothing other than a CNN with two convolutional layers and one fully connected layer, with batch normalization and leaky ReLU activation between layers. We transpose the convolution layers and swap the order to form our Generator, and reverse again to get a conditioning CNN, thus the latter's convolutional network structure is identical to the discriminator.

The network architecture diagram is shown below, with the conditioner network on the top left, the generator on the bottom left and the discriminator on the right. The conditioner takes the previous bar's pianoroll representation as an input to compute feature maps, before concatenating with the noise z and its transposed convolution outputs to generate the pianoroll $G(z)$. The discriminator takes the current time step bar x and $G(z)$ to do the binary classification.



As for training, we applied both the binary crossentropy loss on the discriminator with one side label smoothing, and we also used feature matching loss on the generator. This ensures the distribution similarity on the true and fake data and their feature maps through the first layer of the discriminator. Since it is not easy to pick a set of good hyperparameters, we looked up the official Tensorflow and Pytorch implementation on Github for some hints, but we ended up doing some modifications. The feature loss in MidiNet used the mean square error of x and $G(z)$, but we chose to use the second norm since the original error was too large for the gradients to converge. Also, we constructed 2 convolutional layers instead of 3 on the generator, but increased the number channels to maintain the count of training parameters. A batch size of 32 was applied for 10000 epochs of training. Since the model has only 1.5MB of parameters and not a large image for a piano roll (128 x 16), we used three laptops with CUDA to train on different parameters locally.

Experiment Results, Conclusions and Future Outlooks

We gave our friend the music we generated, and most of them are surprised that the music is generated by computer. Thus, we think our result is quite

successful, but due to the lack of time, a stricter survey could take place to evaluate our music.

Music generation by GAN is interesting and fun, thus we look forward to evolve our model and make more genres of music, such as pop and jazz, which may be harder considering the frequent changes in tempo and melody.

Besides, the MidiNet proposed in the paper had 1D condition vectors to control the chord of the music, which we find difficult to apply on our model at the moment. Since we find it hard to generate an end to our music, the additional constraint may be a useful tool to mark a finale on our generation. Other feature like the type of music might also be controlled. There is potential for further developments, but with limited time and resources, we are satisfied with our finished work.

Reference

MusPy Documentation: https://salu133445.github.io/muspy/

MidiNet Paper: https://arxiv.org/pdf/1703.10847.pdf

MidiNet Official Implementation (Tensorflow): https://github.com/RichardYang40148/MidiNet/tree/master/v1

MidiNet Implementation (Pytorch): https://github.com/annahung31/MidiNet-by-pytorch