Generating Piano Music with Generative Adversarial Network

PAK Ka Yee(55692027), LOW Zhi Hao(54924670)

Background

Real-time music generation is costly and time-consuming to be done by hand. Sound generations will open a new dimension for incredible experiences. With the development of Generative Adversarial Network(GAN), this process just be shortened to just a few seconds. In this project, we hope to simulate piano music with instrumental-only kinds of music.

Objective

1. Compose realistic and enjoyable piano music.
2. Real-time adjustment of the music tunes, rhythm and loudness.

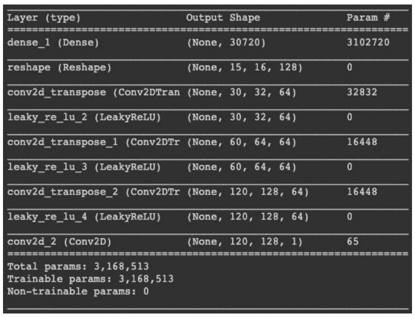
Dataset

We obtained the dataset from Kaggle with Classical Music Musical Instrument Digital Interface (MIDI) by Soumik Rakshit MIDI, which describes music in tracks. Each track has a time and tone dimension. However, each MIDI has a different tempo at different times, making the complexity of the application high. Therefore we decided to use 120 beats/second. Furthermore, we segment the data into 120 beats x 128 tones.

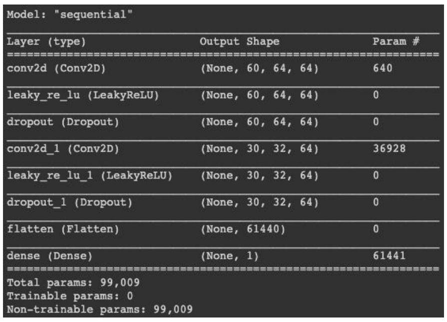
Model Design

There are two agents in GAN, and the first is the generator. The generator takes in the random vector of length 100 and outputs the MIDI 2D matrix using a transposed convolutional neural network. The output of multiple channels is then combined into one channel, the same as the dataset.

The second the discriminator. The discriminator goal is to identify if the input is real or generated from the generator. The discriminator is, therefore, a simple classifier. The loss employed in both models is binary cross-entropy, and the optimizer is adam. We will train the discriminator as usual, and for the generator, we will freeze the discriminator and train on the GAN network to update the weights. Below shows the design of the GAN, left is the generator, right is the discriminator.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 30720)             3102720
_________________________________________________________________
reshape (Reshape)            (None, 15, 16, 128)       0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 30, 32, 64)        32832
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 30, 32, 64)        0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 60, 64, 64)        16448
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 60, 64, 64)        0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 120, 128, 64)      16448
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 120, 128, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 120, 128, 1)       65
=================================================================
Total params: 3,168,513
Trainable params: 3,168,513
Non-trainable params: 0
_________________________________________________________________
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 60, 64, 64)        640
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 60, 64, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 60, 64, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 32, 64)        36928
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 30, 32, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 61440)             0
_________________________________________________________________
dense (Dense)                (None, 1)                 61441
=================================================================
Total params: 99,009
Trainable params: 0
Non-trainable params: 99,009
```

Result

We generated a piece of music for 1 second as an example. We then show the convergence of accuracy on discriminator. As we can see in Epoch 9 and 10, the discriminator starts to converge to 50% accuracy.

Conclusion

A Generative Adversarial Network can be used to generate music, and Real-time music generation can be done by tuning the latent variable. The music melody will follow the tuning of the latent variable to change, following the desired result slowly.

Appendix

YouTube Video Presentation: https://youtu.be/BfAzodgo3Do

Code, Dataset, and Poster:

https://drive.google.com/drive/folders/1ztZSFwE3kfvUH_49zhXrVTVT4gc4Bq3N?usp=sharing

Preproposal and References can follow this link:

https://docs.google.com/document/d/1x6tWYcNPoUouqfQcjisi2dpBjZ1giDQ2ItnqpKlhVfQ/edit?usp=sharing