Creating Chinese Words by GAN Model

National Taiwan Normal University 110-2 Neuron Network,

Assignment 2

Chia-Hao, Chiang
NTNU CSIE Data Visualization Lab
Taipei, R.O.C
61047061s@gapps.ntnu.edu.tw

***Abstract*—Generative Adversarial Network is one of the most symbolic model in machine learning field. The main idea is to let two models: generator and discriminator compete with each other, while generator's goal is to fool the discriminator as taking the generated data as the real one. It's also been used to generate data from the scratch for further usage, and another usage is that implementation of simulation such as climate forecast or cosmological universe data which are hard to observe and are sometimes costly.**

Here, I simply use the dense layers to construct both generator and discriminator, and the goal is to generate non-existing Chinese words.

Keywords—machine learning, generative adversarial network

I. INTRODUCTION

As one of the famous person in AI field, Yann LeCun, once said in the Quora interview:

"This (GAN), and the variations that are now being proposed is the most interesting idea in the last 10 years in machine learning, in my opinion."

The main idea is to train two networks, discriminator and generator, where the former one is used to distinguish whether the input image is real or fake (in binary case, but it can also be a classification problem), while the later one is producing a fake image for discriminator with a random input, i.e. noise. As the training process goes, we expect the images generated by generator can fool the discriminator that take them as real ones, in other words, images produced by generator are having similar distribution to the real ones' in some higher dimension. In addition, the training process is slightly different from normal neuron network, saying that we train discriminator with the real data first, and disable the back propagation (keep the hyper-parameters of discriminator to be fixed) when we next train the generator.

Eventually the fake data generated by generator can pass through discriminator's eyes as the real ones, which can be used in other domains for a kind of data augmentation. One of the famous examples is fake faces generation. The other potential application might be in medical images in my opinion since those images are few and need a lot of time to collect and label.

II. METHODOLOGY

A. Dataset

Offered by National Development Council, the dataset collects 100,000 Chinese-word images, with the size of 24x24x1.

B. Dense Layers

In this project, I simply apply the simplest fully forward connected layer, that is dense layer with Keras. On one hand, I was struggling with the input and output shape when trying to implement CNN networks, which eventually given up wasting time, but directly applying a smaller structure. Although the effects will be worse than convolutional networks, on the other hand, I'd like to see the effectiveness performed by such trivial architecture.

C. Loss Function

We take binary cross entropy in this case since the problem is determining whether the image is fake or real, where $\boldsymbol{N}$ is the number of input samples, $\boldsymbol{y}$ is the prediction of discriminator, and $\boldsymbol{p}$ is the probability of real, while $\boldsymbol{1-p}$ is probability of fake.

$$-\frac{1}{N}\sum_{i=1}^{N}(y_i log p_i + (1-y_i)\log(1-p_i)) \qquad (1)$$

III. EXPERIMENT AND ANALYSIS

Experiments on different number of dense layers are performed. I'd like to observe that if the more layers in generator, the smarter it is, and compare the loss and the accuracy of discriminator of these two.

Moreover, I set the epoch equals to 10,000 times from the fact that after 0.1 million epochs, which takes about four hours, the discriminator loss and accuracy don't change a lot. By this, it will save time in training, but enough for comparing the two different structure. Both are using:

1. Leaky-Relu with alpha = 0.2, which control the underlying value to which the function saturates negatives network inputs.
2. Batch normalization with momentum = 0.8 to speed up the training process.
3. Adam optimizer.
4. 1D Noise input as the size = 128 different values.
5. Batch size = 64.

A. Analysis in Loss and Accuracy

The two models have different structure as shown in table1 below.

Input size: 24x24x1	Generator		Discriminator	
	Model A	Model B	Model A	Model B
Number of layers	5	3	3	2

Number of nodes	256 512 1024 512 256	256 512 1024	512 256 128	512 256

Table 1. comparison of the two model structure. Model-A has more hyper-parameters than Model-B.

- Observation of Discriminator Loss.

The discriminator loss is the average of the one when trained by the real images and one trained by generated images (fake data). As shown in Figure 1.1, this metrics indicates if the discriminator distinguishes the input image correct. The higher the losses, the more mistakes discriminator makes, since the loss from real images are not likely to be significantly different because we are using the same real dataset. Aa we can see, in Model-A, the line zigzags, showing the struggling of discriminator with telling the input image is true of false, while in Model-B, the loss is basically lower and fluctuate very little, meaning that in most of cases, the discriminator successfully classify the inputs.

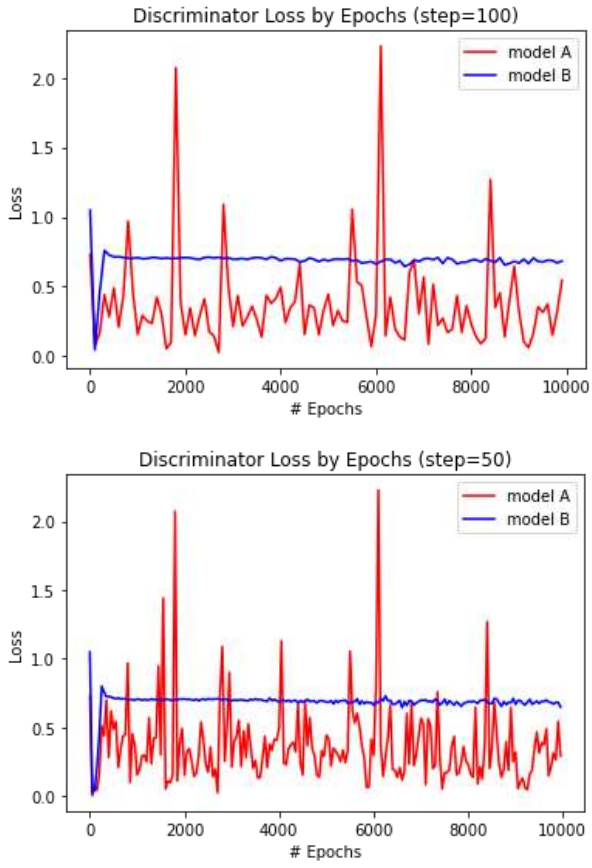


Figure 1.1 the line plots show the loss of discriminator by every 50 (above) and 100 (below) steps of epochs for better graph-telling. We can clearly see the zigzags of Model-A, which has more layers and nodes.

- Comparison of Discriminator Accuracy

Similarly, let's look the accuracy that discriminator performs. The higher the accuracy, the smarter the discriminator is. As we can see in Figure 1.2, Model-A has averagely higher accuracy than Model-B.

- Comparison of Generator Loss

After comparing the discriminators in both models, now we move on to the performance of generators. This metrics tell the ability of generator for fooling the discriminator, as shown in Figure 1.3. The higher loss is, the less likely for generator to mislead the discriminator. We can tell that, in average, Model-A has higher losses; however, in a competing sense, the loss should be zigzag, saying that the generator learns another tricks for a success of fooling the discriminator, while in Model-B, although the losses are lower but not in the form of zigzag, this informs the discriminator is too weak, causing the generator loss rapidly to converge.

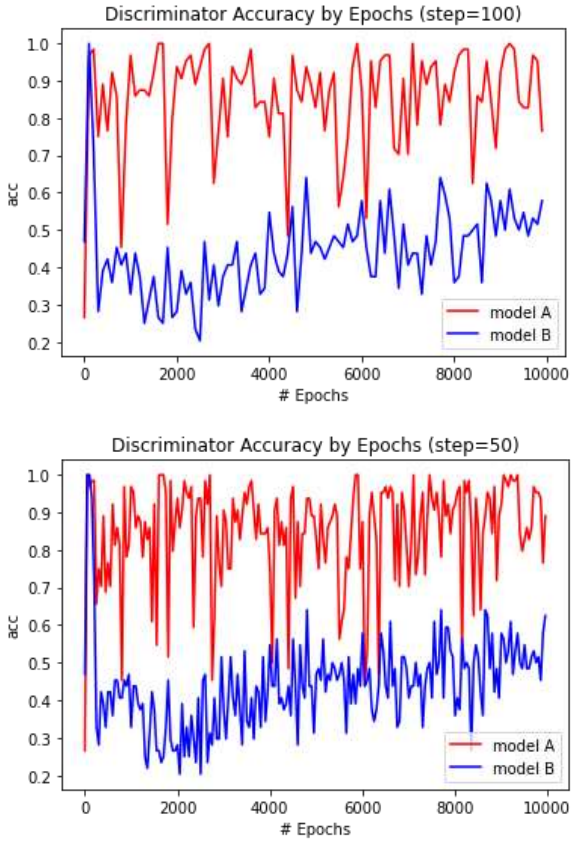
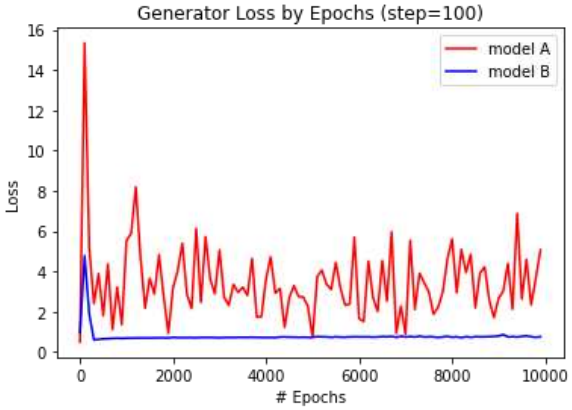


Figure 1.2 the line plots show the accuracy of discriminator by every 50 (above) and 100 (below) steps of epochs for better graph-telling. We can clearly see the zigzags of Model-A, which has more layers and nodes. It shows that discriminator of Model-A performs better than Model-B's.



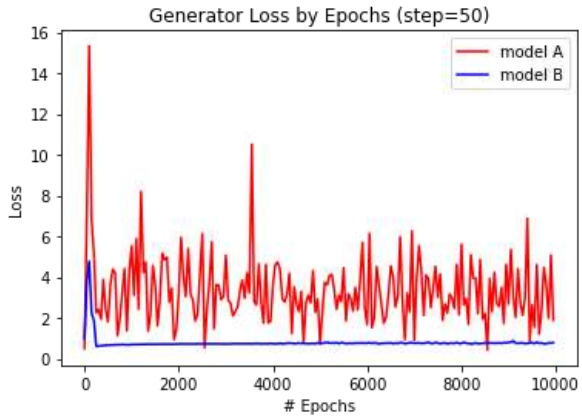


Figure 1.3 the line plots show the loss of generator by every 50 (above) and 100 (below) steps of epochs for better graph-telling. We can clearly see the zigzags of Model-A, which has more layers and nodes. It shows the competition behavior between the generator and the discriminator keeps through the whole training process, while Model-B's line converges early, which implies the unbalanced ability between them.

B. Generated Results of Chinese Words

The generated images are saved every 1000 epochs. At first, the input is pure noise. In the 1000th result, Model-A seems to generate some repeated segment, while Model-B is relatively still random looking, as show in **Figure 2.1**.

In 2000th result, Model-A still generates some similar images with few curves and short lines, while we can see some longer lines forming in Model-B, as shown in **Figure 2.2**.

In 3000~5000 epochs, Model-A starts to generate some line-shape segment in the images but only locates at left side, while Model-B seems to have little changes, as shown in **Figure 2.3**.

After 5000 rounds, Model-A outputs blurred images comparing to the previous ones, but more concrete lines after 7000 epochs. While Model-B outputs only several discrete and scattered short line segments, as shown in **Figure 2.4**.

In the end, compare to the real dataset, the outputs of both models are not acceptable or recognized as words in a basic standard of continuous line or curve segment, or even the whole-white background color is not achieved. The main problem is due to the layers applied in this assignment are purely dense layers. As many papers suggest, it's would be more effective to use convolution network to achieve better outputs.

In addition, I also trained Model-A for 0.1 million times but it turned out to be the same result of blurred images, or even worse, remaining only repeatedly distributed points, not even giving a line. Therefore, I only train for 10,000 times to save the time.

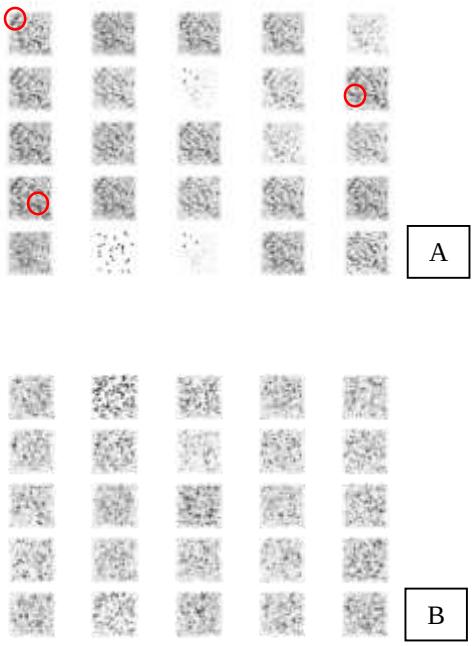


Figure 2.1 the generated images in the 1000th epochs. We can observe some short line forming in Model-A above, while below, Model-B is still random-looking.

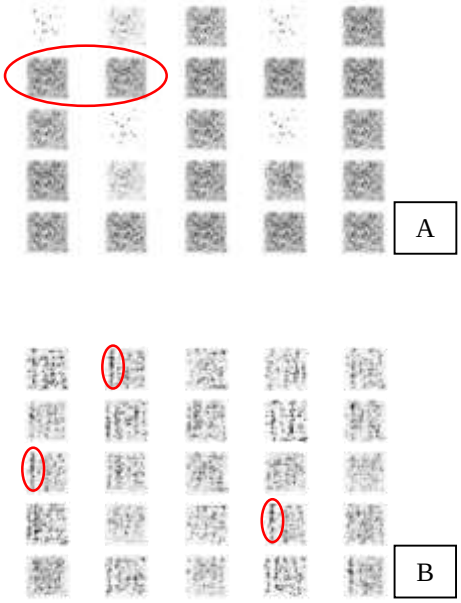


Figure 2.2 the generated images in the 2000th epochs. We can observe some repeated pattern in Model-A above, while below, Model-B starts to produce line shapes.

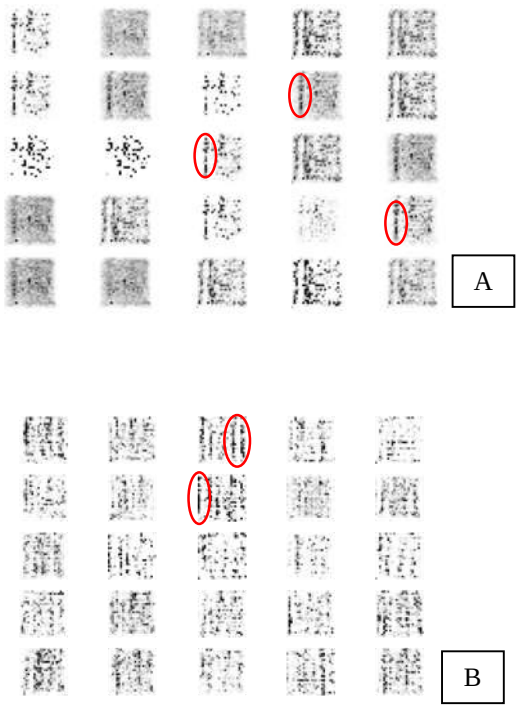


Figure 2.3 the generated images in the 4000th epochs. We can observe line segments generated at left side in Model-A above, while below, Model-B produce more line shapes.

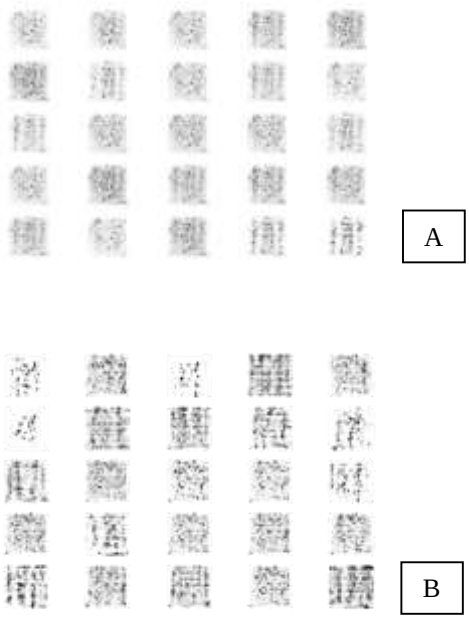


Figure 2.4 the generated images in the end. The output images generated by Model-A shown above becomes blurred, while below, Model-B also becomes blurred. Both models also put background color into more gray rather than white, which are not able to form a single word.

IV. Conclusion

After training with different values of batch size and epoch, we could conclude that:

1) As number of layer increases, the performance is more likely to realize the core behavior of competing with each other, but it requires more time training.

2) Dense layer architecture might not be a proper one for generating images, as seen in the previous part.

V. Future Work

First, we only provide dense-layer-based structure of models, while it's more appropriate by applying convolution network when dealing with images.

Second, the design of number of layers is another adjustable part since it's obviously showing how complicated the network is. However, it sometimes comes out to be a trial and error result, where people genuinely don't know exactly what the machine learns. And perhaps, change others like activation functions and

Moreover, there are plenty of pre-trained model nowadays, and it would be more adequate to convince other people how the model performs by comparing with other ones.

VI. Acknowledgment

Thanks to Professor, Yeh and TAs' sample code for the GAN model. It's quite interesting and astonishing to tune different parameters and expect different outputs, which are literally unpredictable for us, but mysterious. Although the outputs are hardly recognized as a Chinese word, I do observe the changes from pure noises to some organized structure.

What I have learned from this assignment is building a GAN model from the very basics:

1. Arrangement of the training process, since we first train discriminator with the real dataset and then generator, but freeze discriminator.
2. The sizes of given by the generator and the sizes taken and given by the discriminator, and I did struggle with this.
3. Make the use of Google Colab, since it supports GPU and does train faster than on my laptop. However, I should have learned how to install and set the CUDA toolkit on my own.

VII. References

1. Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).