PAPER • OPEN ACCESS

Generation of Handwritten Numbers Using Generative Adversarial Networks

To cite this article: Zhaoxiang Qin and Yuntao Shan 2021 *J. Phys.: Conf. Ser.* **1827** 012070

View the article online for updates and enhancements.

The Electrochemical Society
Advancing solid state & electrochemical science & technology
240th ECS Meeting
ORLANDO, FL
Orange County Convention Center
Oct 10-14, 2021
Abstract submission due: April 9
SUBMIT NOW

This content was downloaded from IP address 158.46.128.255 on 17/03/2021 at 13:29

Generation of Handwritten Numbers Using Generative Adversarial Networks

Zhaoxiang Qin[1, a, †] **and Yuntao Shan**[2, b, †]

[1]Department of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, China;

[2]Suzhou Science & technology town foreign language school, Suzhou, Jiangsu, China.

[a]3180106169@zju.edu.cn, [b]luchengbinhenry@gmail.com

[†]These authors contributed equally.

Abstract. As a promising generative modeling method, Generative Adversarial Networks are a deep-learning-based generative model, in which two networks, namely the generative network and the discriminant network, contest with each other in a game during which the generator "learns" and get trained to be able to fool the discriminator of believing a specific image is (superficially) authentic. It is widely used to generate pictures based on existing data. Based on the MNIST data set, this paper studies and analyzes the application of Generative Adversarial Networks in the generation of handwritten digital images. In addition, by studying the Deep Convolutional Generative Adversarial Network and the Conditional Generation Adversarial Network model, we compared the connections and differences between these three models in training the MNIST dataset to generate handwritten digits algorithms, and gave an optimization method for generating handwritten digits.

1. Introduction

As a deep learning model, Generative Adversarial Network (GAN) was first proposed by Ian J. Goodfellow and others in the article "Generative Adversarial Networks" in October 2014. The Generative Adversarial Network was described as a framework for estimating generative models through adversarial processes. The framework realizes the game between the Generative Model (G) and the Discriminant Model (D) by simultaneously training G that captures the data distribution and D that estimates the probability of the sample from the data, and finally makes G reproduce the distribution of the training data and make the probability value in D equal to 0.5. The result of the game is to make the discriminant model unable to judge whether the output result of the generative model is true.

The Generative Adversarial Network can generate data that cannot be distinguished true or false based on existing data, so its most direct application is image generation. Researches in recent years have shown that the application of generative adversarial network has been further deepened. People are no longer satisfied with generating images based on existing data but hope to achieve a text-to-image process, that is, by adding descriptive sentences which will provide more flexibility and directivity for image generation. For instance, there already exists research on translating text in the form of single-sentence, human-written descriptions directly into image pixels [1, 2] and Progressive Semantic Image Synthesis [3].

Based on previous literature, it's observed that there are countless studies on the use of generative

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

adversarial network models to generate images, but most of them are based on one most common generative adversarial network model – Deep Convolutional Neural Network- to implement and evaluate the results. For example, "ImageNet Classification with Deep Convolutional Neural Networks" [4] uses deep convolutional neural networks to classify up to 1.2 million high-resolution images. In fact, there are many types of generative adversarial network models, and they all optimize the most basic generative adversarial network model to a certain extent according to the expectations of researchers. However, only few researches on generative adversarial network models compare and analyze the results of image generation of different models based on the same data set. The purpose of this article is to fill the scarcity of research in this field. Therefore, this paper analyzes and compares the training results of three different network models of GAN, Deep Convolutional Generative Adversarial Network (DCGAN), and Conditional Generation Adversarial Network (CGAN) based on the most basic MNIST dataset to generate handwritten digital images.

2. Method

In this section, we will mainly introduce the MNIST dataset, GAN, DCGAN, CGAN and the preprocessing of training data.

2.1. MNIST dataset

Mixed National Institute of Standards and Technology, also known as the MNIST dataset, is a large handwritten digit database collected by the National Institute of Standards and Technology. It contains a training set of 60,000 examples and a test set of 10,000 examples. The dataset of handwritten digits consists of a training sample set with a buffer size of 60k, and the buffer size has a batch size of 256 to calculate the shape of an image with a size of 28 × 28 × 1. The dataset is centralized and separated from the boxes of handwritten digital images [5].

Here we use TensorFlow to display the first 20 sample images in the training set. As is shown below in Figure 1.

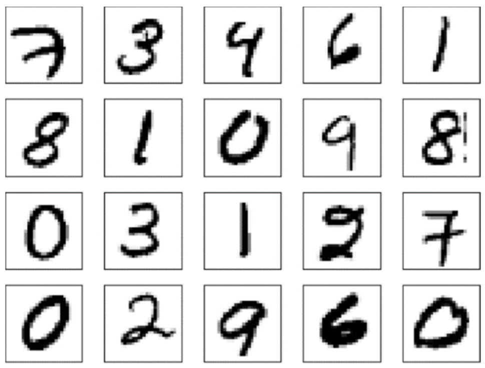


Figure 1. Sample images in the training set of MNIST dataset

2.2. GAN

The Generative Adversarial Network was first proposed by Goodfellow et al. in 2014 [6]. It consists of a pair of models called the generator and discriminator. The generator uses the discriminator as a loss function and updates its parameters to generate more realistic-looking data. On the other hand, the discriminator updates its parameters to better identify fake data from real data. In fact, the generative model can be regarded as a group of thieves trying to generate counterfeit money, and the discriminative model can be regarded as the police trying to detect the counterfeit money thief [7]. This cat and mouse game continues until the system reaches the so-called balance. After reaching the balance, the data created by the generator looks real enough, so the discriminator can only randomly guess whether the generator data is true or false. Therefore, the entire framework is like a two-person minimax game, where the generator tries to minimize its objective function, and the discriminator tries to maximize its objective function. The objective of the game are as follows:

$$\min_G \max_D V(D,G) = e_{x\sim P_{data}e}[\log D(e)] + E_{z\sim P_z(z)}[\log(1 - D(G(z))] \quad (1)$$

According to the formula, we have data x with p_{data} distribution, and noise input z with p_z distribution. The generator is a differential function $G(z;\theta_g)$ represented by a multi-layer perceptron with a parameter of θ_g, where the parameter of the neural network θ_g maps elements from the noise distribution to the target distribution; the discriminator is represented by $D(x;\ \theta_d)$. Similarly, $D(x)$ gives the probability that x comes from the data instead of p_g. D is trained to maximize $\log D(x)$, and G is trained to minimize $\log(1 - D(G(z)))$ [7, 8]. The overall architecture of the adversarial network is shown below in Figure 2.

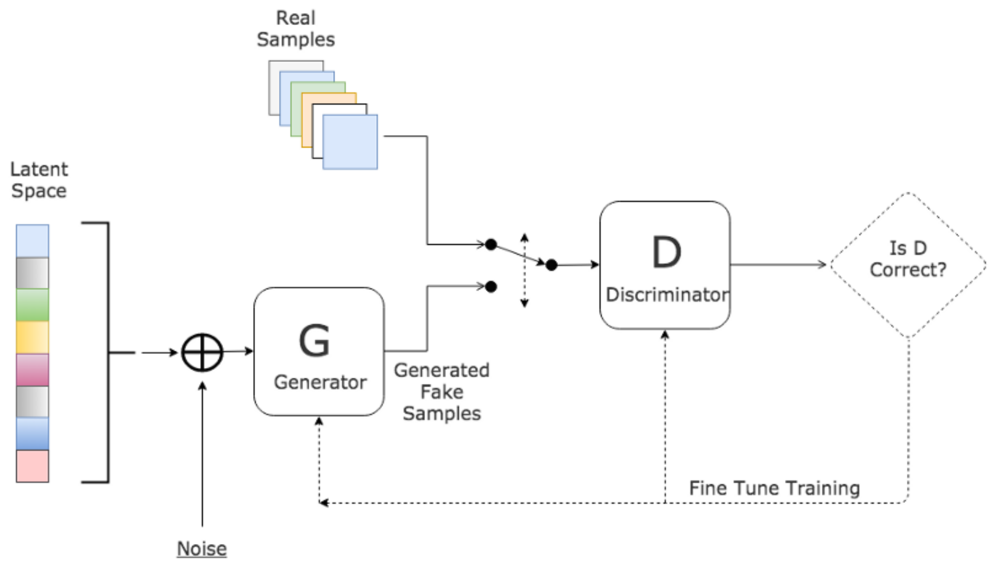


Figure 2. The Structure of the Generative Adversarial Network

The development of traditional generative models is very slow, mainly because of the following two dilemmas: first, modeling the real world requires a lot of prior knowledge, and the quality of modeling directly affects the performance of the generative model; second, real-world data is often very complex, so the amount of calculation required to fit the model is often very huge, even unbearable. However, GAN avoids the above problems through the game of two networks, which is the advantage of GAN compared with the traditional generative model.

However, GAN also has shortcomings. First, G may tend to generate more high-scoring but similar data during the training process, with a lack of diversity, which leads to the so-called model collapse; second, there is still a lack of a standard evaluation system for GAN; third, the traditional GAN training is unstable, the generation process is uncontrollable, and it is not interpretable. Therefore, we need to introduce DCGAN, CGAN, and other advanced models that were designed to optimize the traditional GAN.

2.3. DCGAN

In order to be well adapted to the convolutional neural network (CNN) architecture, DCGAN proposes four-point architecture design rules: use convolutional layers to replace the pooling layer; remove the fully connected layer; use batch normalization; use appropriate activation functions.

2.3.1. Convolutional layer replaces the pooling layer

The traditional CNN structure not only includes a convolutional layer, but also a pooling layer. In the DCGAN structure, the operation of removing all the pooling layers in the traditional convolutional network is adopted, and the convolutional layer is used to replace it. In the discriminator, a strided convolution is used to replace the pooling layer; in the generator, fractional-strided convolutions are chosen instead of the pooling layer.

2.3.2. Remove the fully connected layer
The disadvantage of the fully connected layer is that there are too many parameters. When the number of neural network layers is deep, the operation speed becomes very slow. In addition, it can easily lead to overfitting. A compromise solution has been proposed, that is, the random input of the generator is directly connected with the convolutional layer feature input, and the output layer of the discriminator is also connected with the output feature of the convolutional layer [9].

2.3.3. Use batch normalization
The neural network of deep learning has many layers, and each layer will change the distribution of output data. As the number of layers increases, the overall deviation of the network will become larger. Batch normalization can solve this problem. By normalizing the input of each layer, it can effectively make the data obey a fixed data distribution.

2.3.4. Design of activation function
In DCGAN, the generator and the discriminator use different activation functions. The generator uses the ReLU function, and the output layer uses the Tanh activation function. In addition, LeakyReLU is used for all layers in the discriminator.

In Figure 3 below, we show the architecture of generator in DCGAN.

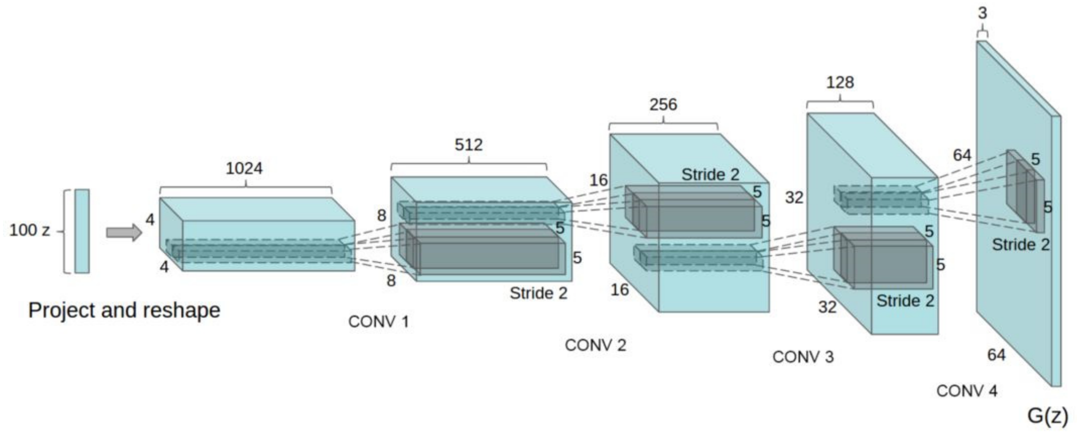


Figure 3. The architecture of generator in DCGAN

In summary, compared with GAN or ordinary CNN, the improvement of DCGAN includes the following aspects: 1) Use convolution and deconvolution instead of pooling layer; 2) Batch normalization operations are added to both the generator and the discriminator; 3) The fully connected layer is removed, and the global pooling layer is used instead; 4) The output layer of the generator uses Tanh activation function, and the other layers use RELU; and 5) All layers of the discriminator use Leaky ReLU activation function.

2.4. CGAN
Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y can be any auxiliary information, such as category tags or data from different forms. We can send y as an additional input layer to the generator and discriminator to add conditions [10].

Given certain conditional information, the objective function of the model becomes the following form:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \quad (2)$$

In the generator model, the conditional variable y is actually used as an additional input layer, which is combined with the generator's noise input $p(z)$ to form a joint hidden layer expression; in the discriminator model, y and the real data x are also input into a discriminant function. In fact, it is to concat z and x with y respectively, as the input of the generator and the discriminator, and then train. In the supervised DBN, a similar approach is also used. The basic framework of Conditional Generative Adversarial Networks is as follows in Figure 4.

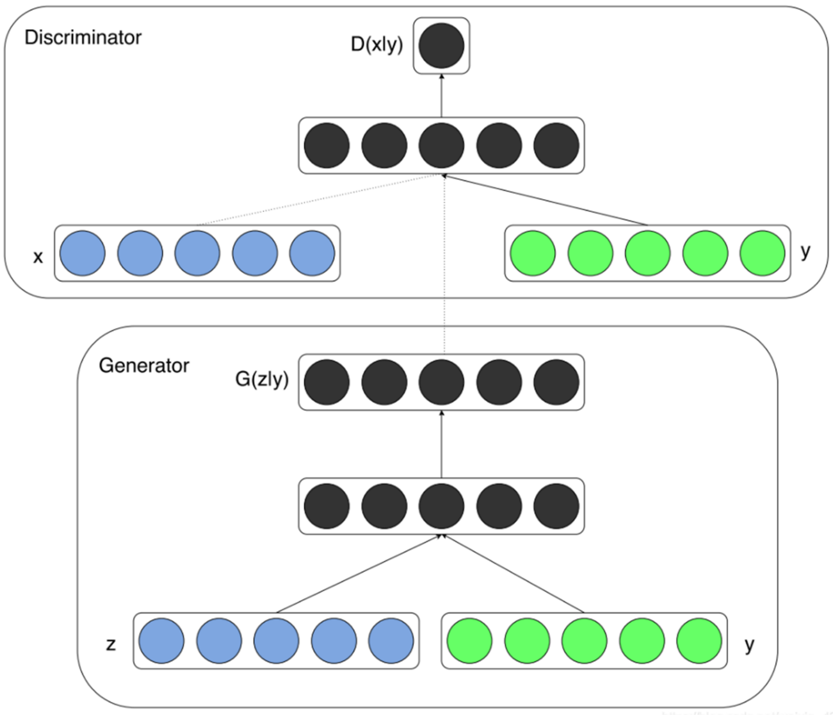
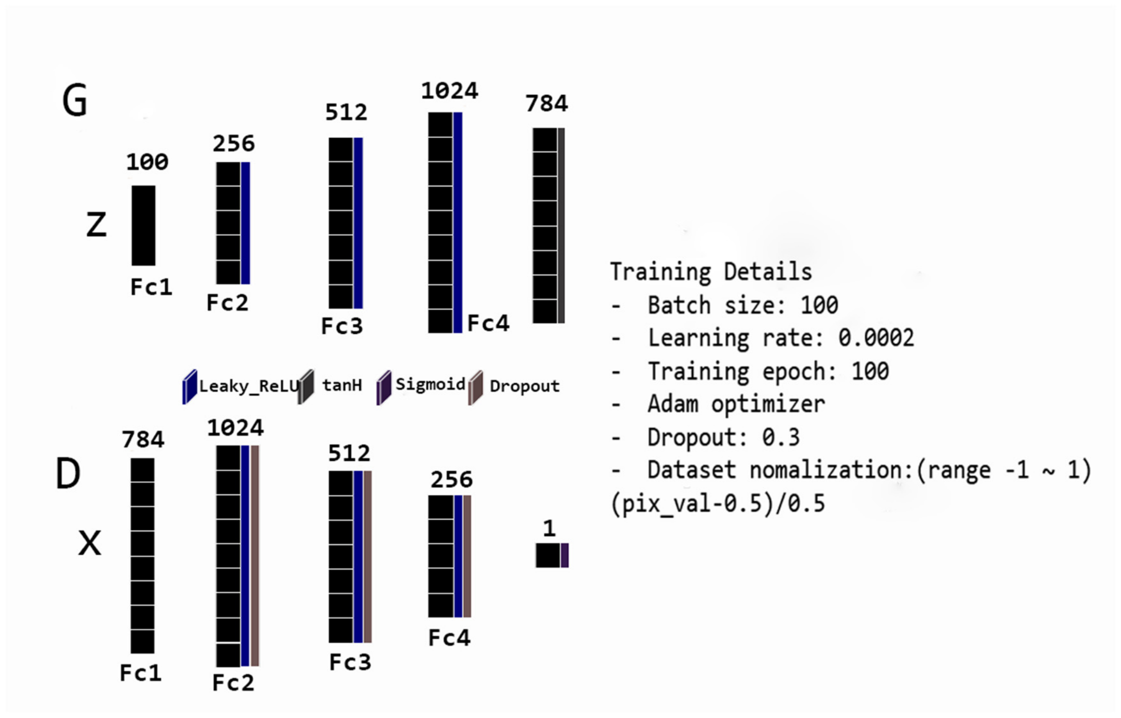


Figure 4. The architecture of conditional GAN(CGAN)

2.5. Training details for various GANs

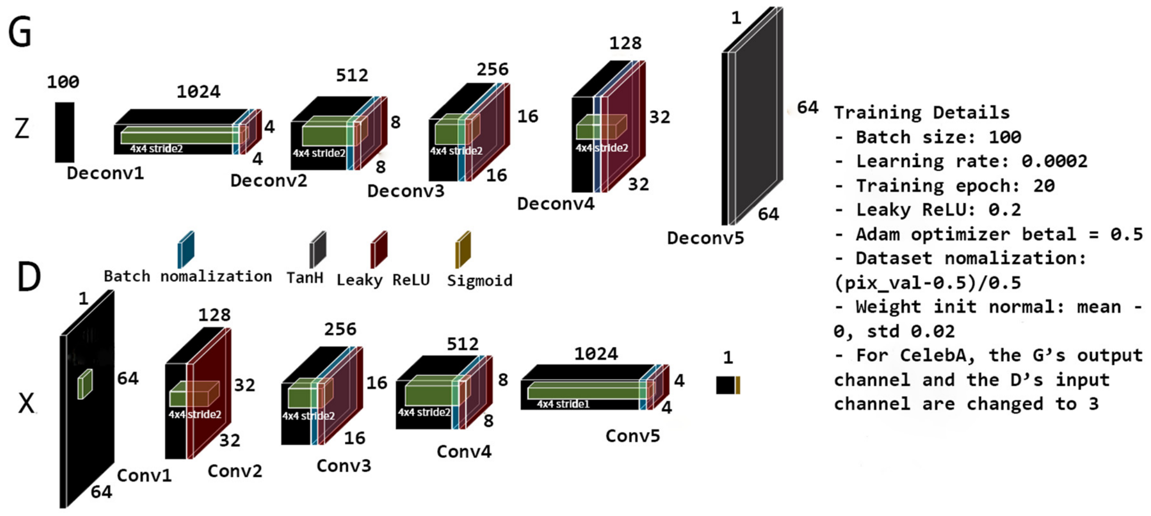
In this subsection, we will mainly determine some basic parameters during the training process, like the batch size, learning rate, training epoch, the optimizer we use. They are shown below in Figure 5-7.

Here we will briefly introduce the parameters below. The meaning of batch size, learning rate and training epoch is obvious, we will not discuss here. In GAN, we will selectively discard some points according to the probability, and dropout represents this probability; in DCGAN, Leaky ReLU represents the activation function, Adam Optimizer beta1 is a parameter in Adam Optimizer; in CGAN, we use the Xavier initialization method to initialize the weights.



Training Details
- Batch size: 100
- Learning rate: 0.0002
- Training epoch: 100
- Adam optimizer
- Dropout: 0.3
- Dataset nomalization:(range -1 ~ 1)
(pix_val-0.5)/0.5

Figure 5. Training details of GAN



Training Details
- Batch size: 100
- Learning rate: 0.0002
- Training epoch: 20
- Leaky ReLU: 0.2
- Adam optimizer betal = 0.5
- Dataset nomalization:
(pix_val-0.5)/0.5
- Weight init normal: mean -
0, std 0.02
- For CelebA, the G's output
channel and the D's input
channel are changed to 3

Figure 6. Training details of DCGAN

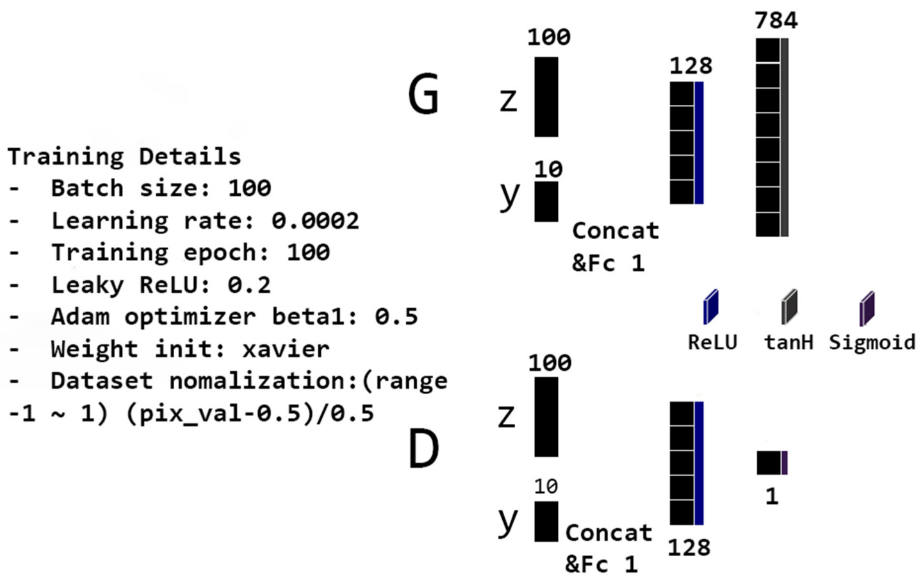

```
Training Details
-  Batch size: 100
-  Learning rate: 0.0002
-  Training epoch: 100
-  Leaky ReLU: 0.2
-  Adam optimizer beta1: 0.5
-  Weight init: xavier
-  Dataset nomalization:(range
-1 ~ 1) (pix_val-0.5)/0.5
```

Figure 7. Training details of CGAN

3. Experiment and Results

We conducted preliminary experiments using the MNIST dataset. We do not use a handwritten character data set, rather a handwritten digit database to test the range of architecture for generating numbers. We accordingly use the ASCII values of the numbers 0 to 9 for this purpose. Extensive changes in writing style can lead to changes in the generated images of the same number. For specific writing styles, we believe that this change is small, and the generator network will be more accurate when copying specific writing styles [11].

For GAN, we set the training epoch to 100 and train the data set with a learning rate of 0.0002 and a batch size of 100. The loss functions of the generator (G_loss) and the discriminator (D_loss) are recorded along with the training epoch. The handwritten digital picture generated when the training epoch is 100 is generated and compared with the raw pictures of the MNIST data set, which are recorded as shown in Figure 8 and Figure 9 below.

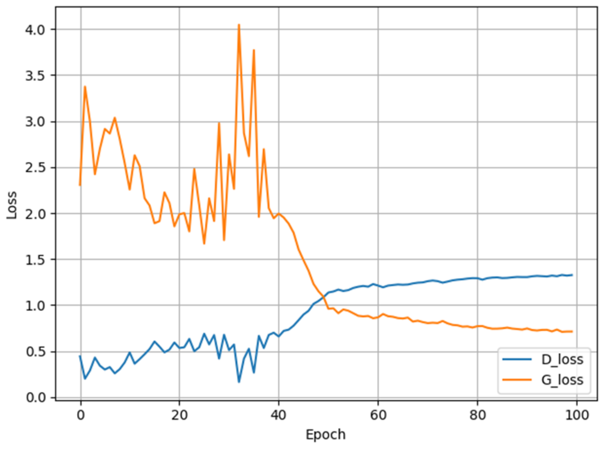
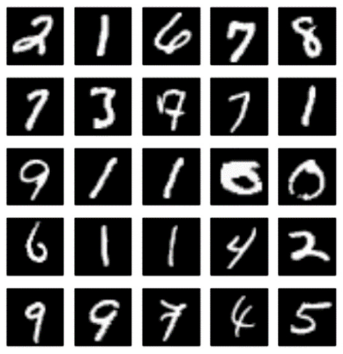
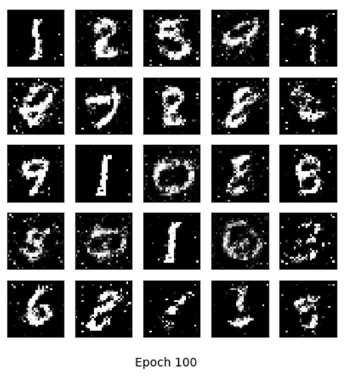


Figure 8. The curve of the loss function with the training epoch





(a) raw image (b) generated image

Figure 9. Raw MNIST image and image generated in epoch 100 with GAN

The understanding of G_loss and D_loss is further explained here: D_real represents the output of the discrimination network for real samples, and D_fake represents the output of false samples. When D is optimal, 1 is output for all D_real and 0 is output for all D_fake, so D_loss at this time is the smallest, that is, it is judged that the loss of network D reaches the minimum. When G is optimal, the output of D_fake should be 1 (because the sample generated by G is close to the real sample at this time, D can no longer judge at this time, and think that the sample generated by G is the real sample), at this time G_loss is the smallest, that is the loss of the generating network G reaches a minimum.

It can be found that after iterating 100 epochs, the effect of handwritten digital images generated by GAN is not particularly good. So, we next show the results of training using DCGAN. In Figure 9, we record the image generated by training 20 epochs using DCGAN.

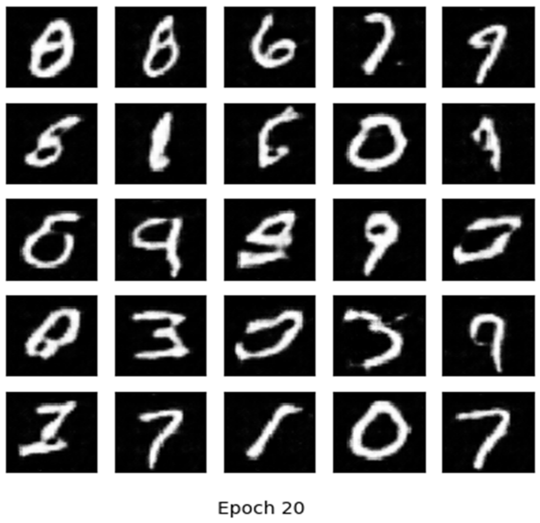


Figure 10. Image generated in epoch 20 with DCGAN

And for CGAN, the results are shown below in Figure 10-1 and Figure 10-2.

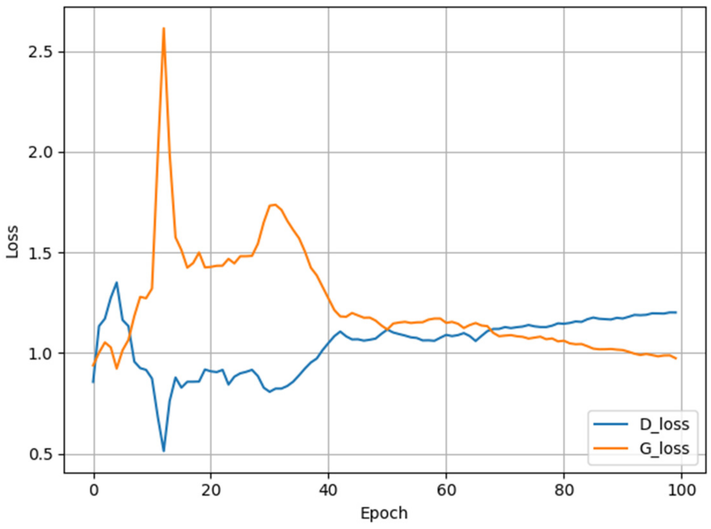
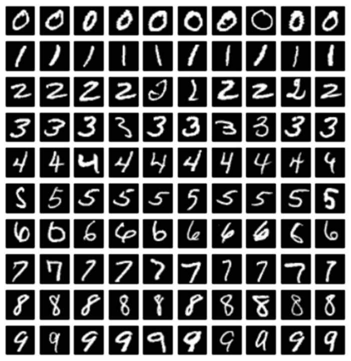
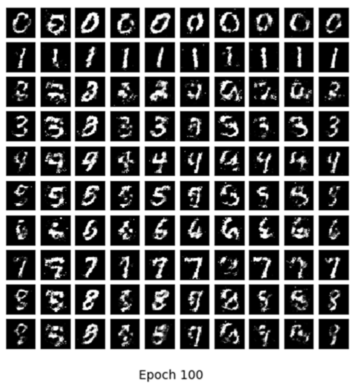


Figure 11. The curve of the loss function with the training epoch

(a) raw image (b) generated image

Figure 12. Raw MNIST image and image generated in epoch 100 with CGAN

4. Conclusion

By comparing the training results of the three neural network models, we found that because DCGAN introduces the convolutional network into the GAN structure and uses the powerful feature extraction capabilities of the convolutional layer to improve the effect of GAN, DCGAN can be generated in less training times with an output of clearer image. The difference between CGAN and GAN is that both the generator and the discriminator have added some additional conditions, such as class tags. The overall optimization process is similar to that of GAN, so the handwritten digital images generated by CGAN and the GAN's clarity and number of training iterations are also roughly the same. Besides, in DCGAN the D and G have to learn to hierarchy to scene the objects, while the CGAN has to maximize the function of G and minimize the function D.

Overall, for DCGAN, the pixels of the generated image are clearer, but the disadvantage is that if the GPU is not powerful enough, the training process is very time-consuming. Besides, optimization is compulsory for DCGAN because the losses in the generator and discriminator are available. While for CGAN, because of the additional parameters added in the CGAN, we have to merge the labels in the discriminator and generator function (5), and the clarity of the generated image is roughly the same as

GAN.

5. Future work

In the process of training to generate handwritten digital pictures, we found that the pictures generated in some epochs did not have the expected pattern of handwritten digital pictures.

After consulting the literature, we learned that this is the result of the so-called model collapse. Take the generation of MNIST handwritten digit images as an example. Assuming that there are 10 different patterns for digits 0-9, and our network can only predict and generate some patterns of digits but cannot recognize other digits, this is a mode collapse.

The optimization scheme for mode collapse is mentioned in many documents. For example, some scholars have proposed a manifold guided generative adversarial network (MGGAN), which leverages a guidance network on existing GAN architecture to induce generator learning all modes of data distribution so as to alleviate the mode collapse. In fact, there are many ways to optimize mode collapse, so we hope to further analyze the optimization methods of mode collapse in the future and compare their advantages and disadvantages in our next study.

References

[1] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee, Generative adversarial text to image synthesis. 33rd Int. Conf. Mach. Learn. ICML 2016. **3**, 1681–1690 (2016).
[2] H. Park, Y. Yoo, N. Kwak, MC-GAN: Multi-conditional generative adversarial network for image synthesis. Br. Mach. Vis. Conf. 2018, BMVC 2018 (2019).
[3] K. Yue, Y. Li, H. Li, Progressive semantic image synthesis via generative adversarial network. 2019 IEEE Int. Conf. Vis. Commun. Image Process. VCIP 2019, 19–22 (2019).
[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks. Commun. ACM (2017), doi:10.1145/3065386.
[5] Prabhat, Nishant, D. K. Vishwakarma, Comparative Analysis of Deep Convolutional Generative Adversarial Network and Conditional Generative Adversarial Network using Hand Written Digits. Proc. Int. Conf. Intell. Comput. Control Syst. ICICCS 2020, 1072–1075 (2020).
[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets. Adv. Neural Inf. Process. Syst. **3**, 2672–2680 (2014).
[7] S. Hitawala, Comparative Study on Generative Adversarial Networks. arXiv (2018).
[8] E. Hernandez, D. Liang, S. Pandori, Generative Models for Handwritten Digits, 1–7.
[9] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks. 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., 1–16 (2016).
[10] M. Mirza, S. Osindero, Conditional Generative Adversarial Nets, 1–7 (2014).
[11] A. Ghosh, B. Bhattacharya, S. B. R. Chowdhury, Handwriting profiling using generative adversarial networks. 31st AAAI Conf. Artif. Intell. AAAI 2017, 4927–4928 (2017).