Fashion Image Generation Using GAN Model

LIU Minghao 21096308D

Abstract

Generative adversarial networks are a class of neural networks used to generate data. The basic GAN model consists of two parts: a generator and a discriminator. The generator generates fake images from random noise. The discriminator tries to distinguish the fake images from the real ones. These two networks are trained alternately. The generator tries to generate more realistic images to trick the discriminator, while the discriminator tries to distinguish the fake images from the real ones. The structure of the GAN model is shown in Figure 1:

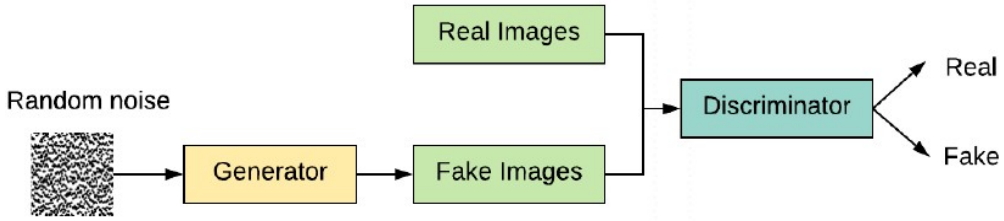


Figure 1. Training process of GAN model

1 Introduction

This project aims to generate fashion images using GAN model. The dataset is from the Fashion-MNIST dataset. The model is trained on the dataset and the generated images are evaluated by human and machine. Users can use the trained model to generate randomly fashioned images with similar resolution to the dataset, or generate corresponding images based on a specific label.

1.1 Dataset

The dataset is from the *Fashion-MNIST* dataset. It is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The program use the dataset as the train loader which use for judging the generated images. The dataset is from Kaggle, and the images are shown in the following figure 2:

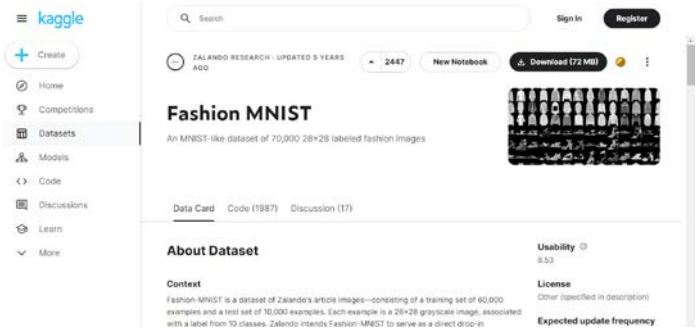


Figure 2. Fashion-MNIST dataset

1.2 Visualisation

The images are 28x28 pixels and each pixel value is stored in the dataset. The program uses "plt" to visualize the dataset in the train loader. It uses a function to reshape a 784-dimensional vector from the train csv into a 28x28 image. And draws 10 images from each label and let all image show in once time. The images are shown in the following figure 2:

Figure 3. Visualization of the dataset

2 Baisc GAN model

2.1 *Model Structure*

As a basic GAN, all layers in the generator and discriminator are fully connected layers. For the generator, the input is a 100-dimensional random noise vector, and the output is a 784-dimensional vector, which is reshaped into a 28*28 image. For the discriminator, the input is a 784-dimensional vector, and the output is a scalar between 0 and 1, which represents the probability that the input image is real.

The activation function of the generator is ReLU, and the activation function of the discriminator is LeakyReLU. The loss function of the generator is the binary cross entropy with logits. The optimizer of the generator and discriminator is Adam. The learning rate is 0.002, and the batch size is 256.

2.2 *Model Performance & Loss*

The model is trained for 100 epochs with 256 batch size. The loss of the generator and discriminator is shown in the following figure 4:

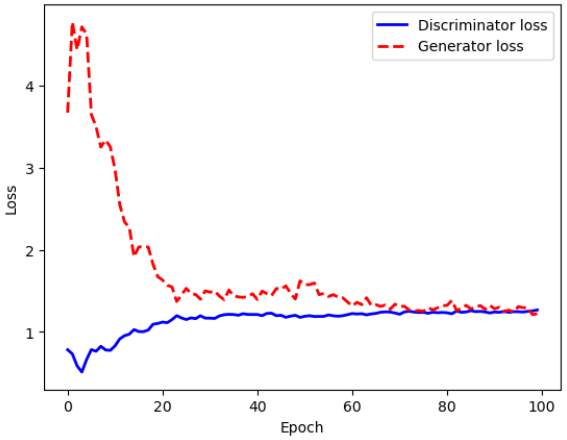


Figure 4. Loss of the generator and discriminator (Basic GAN)

And the generated images during the training are shown in the following figure 5:

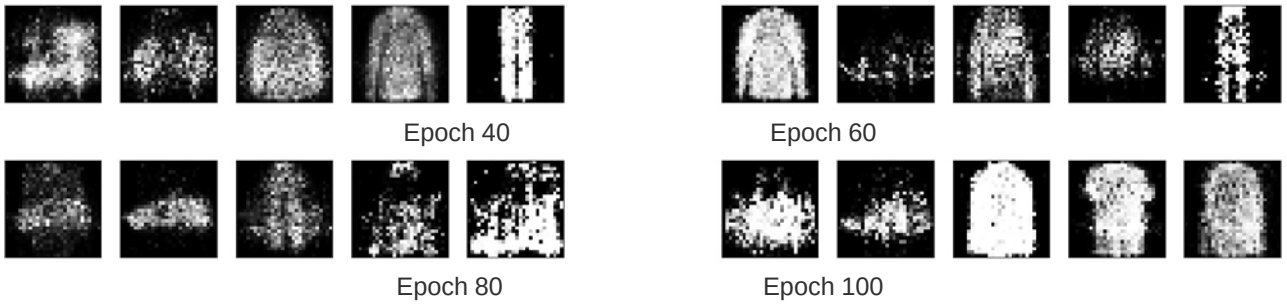


Figure 5. Generated images during the training (Basic GAN)

As we can see from the figure 4, the generated results become clearer as the training epoch increases. However, from epoch 80 to 100, the generated results become worse. As shown in the figure for the generator loss and discriminator loss, the two loss functions are almost the same when the epoch is greater than 70. This means that the generator and discriminator are in equilibrium. The model is over-fitted and generates bad results. This is due to the fact that both the generator and discriminator are fully connected layers. The model is too simple to produce good results.

3 *Optimization GAN Model: cGAN*

3.1 *Model Structure*

In contrast to the base GAN, the generator and discriminator of **cGAN** are no longer simply fully connected layers. Compared with the usual Conditional GAN, our **cGAN** is more inclined to **DCGAN**. Our **cGAN** is based on convolutional layers combined into generators and discriminators. And use tanh as the activation function of the generator. The discriminator uses sigmoid as the activation function. This change will make the images generated by the model closer to the real ones, and the effect is much higher than that of the base GAN model. The model structure is shown in the following figure 6:

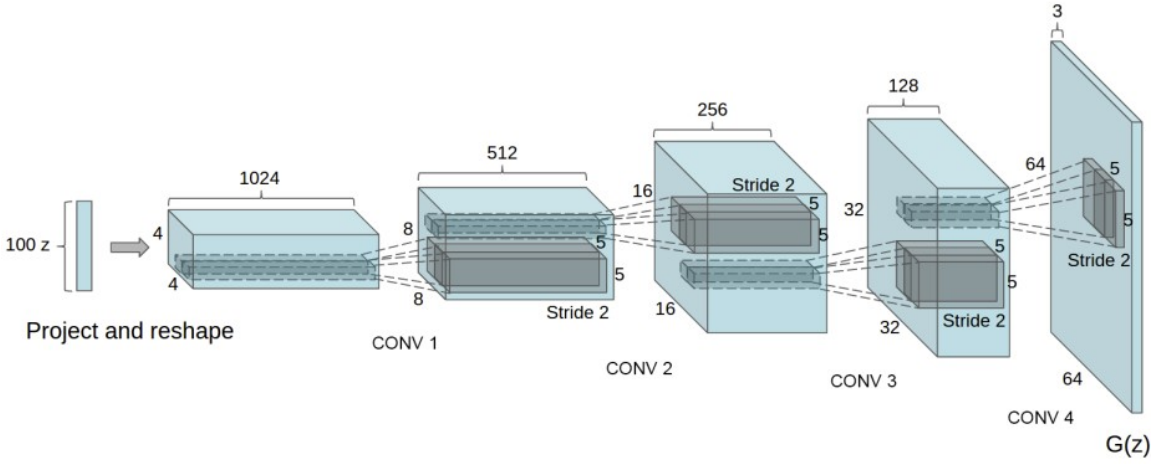


Figure 6. Model structure of cGAN

3.2 *Loss*

The model is trained for 100 epochs with 256 batch size. The loss of the generator and discriminator is shown in the following figure 7:

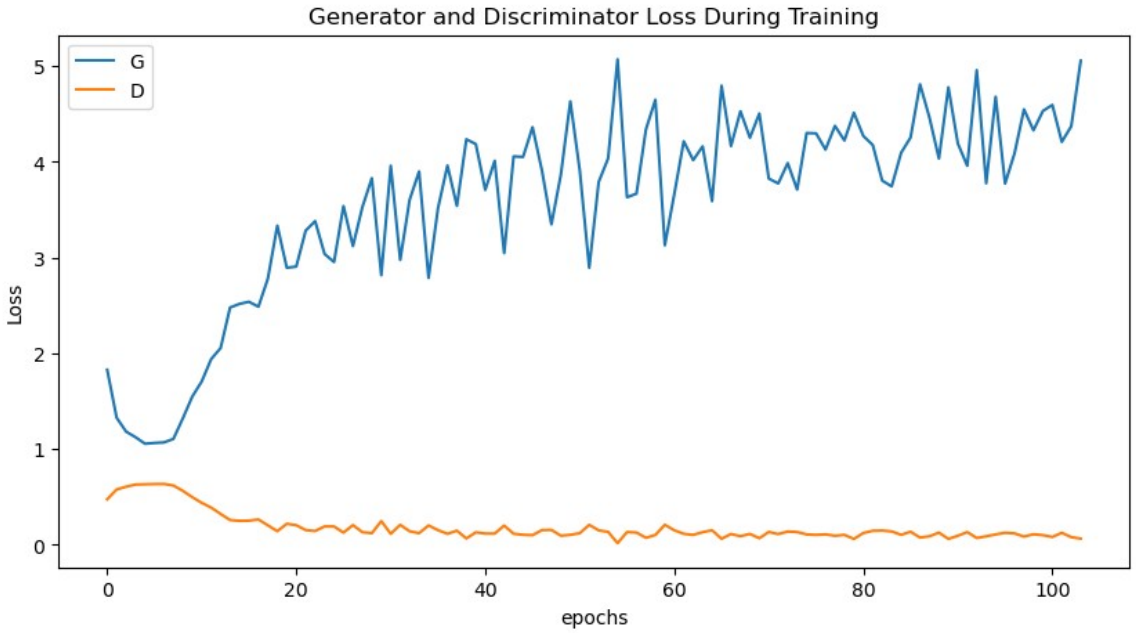


Figure 7. Loss of the generator and discriminator (**cGAN**)

As the discriminator is a binary classifier, the loss function of the discriminator is the binary cross entropy with logits. The loss function is shown in the following figure 8 (From paper "Conditional Generative Adversarial Nets", Author by Mehdi Mirza and Simon Osindero):

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

Figure 8. Loss function (Basic GAN)

By analyzing the loss images, we can see that the losses of the generator and discriminator are getting larger as the number of epochs increases. This result is a good representation of the generator and discriminator **"against"**. The generator tries to generate images that are more and more similar to the real ones, and the discriminator tries to distinguish the real images from the fake ones. This **"against"** produces a loss function that makes the generated images more realistic by back propagation.

3.3 *Generated Images*

The generated images during the training are shown in the following figure 9:

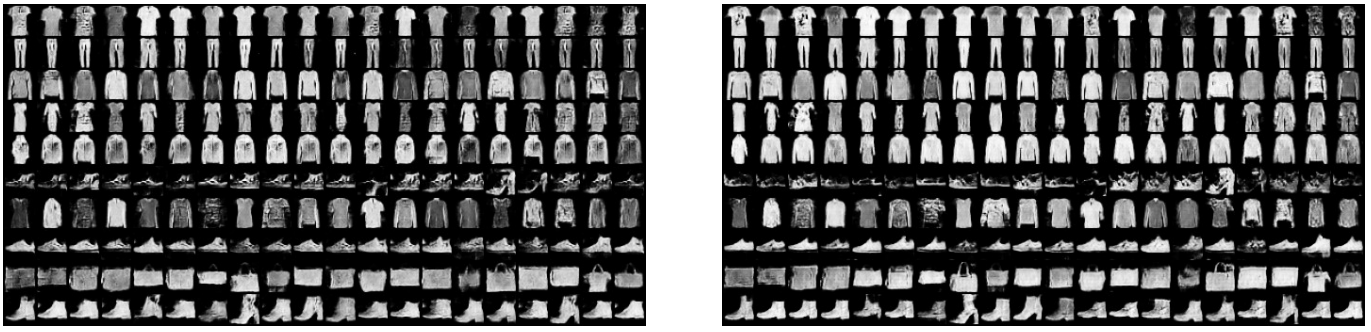


Figure 9. Generated images during the training (cGAN)

From the figure 8, we can see that the generated images are getting clearer as the number of epochs increases. The generated images are very similar to the real ones. This is because the model uses convolutional layers instead of fully connected layers. The model is more complex and can generate more realistic images.

However, the generated images are still not very clear. This is because the loss function using the **"LOG"** function. In this case, the process of finding the generator minimum can be equated to finding the JS scatter distance, but this can easily lead to discriminator saturation, i.e., the backpropagation gradient used to update the discriminator parameter vanishes.

4 Optimization cGAN Model: wGAN & wGAN-GP

4.1 Model Structure

In order to solve the problem of the loss function of **cGAN**, I decive to use **wGAN** and **wGAN-GP** to optimize the model. As these two models do well in Fashion MNIST dataset, the reserch as shown in the following figure 10 (From paper "Are GANs Created Equal? A Large-Scale Study", Author by Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet):

	MNIST	FASHION	CIFAR	CELEBA
MM GAN	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2
NS GAN	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3
LSGAN	7.8 ± 0.6*	30.7 ± 2.2	87.1 ± 47.5	53.9 ± 2.8*
WGAN	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
WGAN GP	20.3 ± 5.0	24.5 ± 2.1	55.8 ± 0.9	30.0 ± 1.0
DRAGAN	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9
VAE	23.8 ± 0.6	58.7 ± 1.2	155.7 ± 11.6	85.7 ± 3.8

Figure 10. Different GANs Results in Different Datasets

It can be seen from the figure 9 that the **wGAN** and **wGAN-GP** models have better results than the **cGAN** model in the Fashion MNIST dataset. Therefore, I decided to use **wGAN** and **wGAN-GP** to optimize the model.

4.1.1 *wGAN*

The main differences between **wGAN** and **cGAN**. The first is the loss function. The loss function of **wGAN** is shown in the following figure 11 (From paper "Are GANs Created Equal? A Large-Scale Study", Author by Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet):

WGAN $\quad \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x \sim p_d}[D(x)] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})] \qquad \mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$

Figure 11. Loss function (wGAN)

Since the loss function of **wGAN** is no longer a **"LOG"** function but a **"MEAN"**, and the sigmoid function, which is prone to gradient disappearance in the generator, is removed, the problem of discriminator saturation can be solved. The another is the weight clipping. The weight clipping is used to limit the weight of the discriminator. The weight I use for weight clipping is 0.01. The weight clipping can make the discriminator more stable.

4.1.2 *wGAN-GP*

The main difference between **wGAN-GP** and **wGAN** is that the loss function of **wGAN-GP** has an extra computation of gradient penalty. The loss function of **wGAN-GP** is shown in the following figure 12 (From paper "Are GANs Created Equal? A Large-Scale Study",Author by Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet):

WGAN GP $\quad \mathcal{L}_{\mathrm{D}}^{\mathrm{WGANGP}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g}[(||\nabla D(\alpha x + (1-\alpha \hat{x})||_2 - 1)^2] \qquad \mathcal{L}_{\mathrm{G}}^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$

Figure 12. Loss function (wGAN-GP)

After each calculation of the discriminator, the model use `hat = alpha x real + (1 - alpha) x fake` to calculate the gradient penalty. The alpha is a random number between 0 and 1, base on the batch size. The gradient penalty is calculated by `gradient penalty = mean[(gradients.norm(2, dim=1) - 1)^2]`. The gradient penalty can make the discriminator more stable than **wGAN** model.

4.2 *Loss*

The **wGAN** model is trained for 100 epochs with 256 batch size. The loss of the generator and discriminator is shown in the following figure 13:

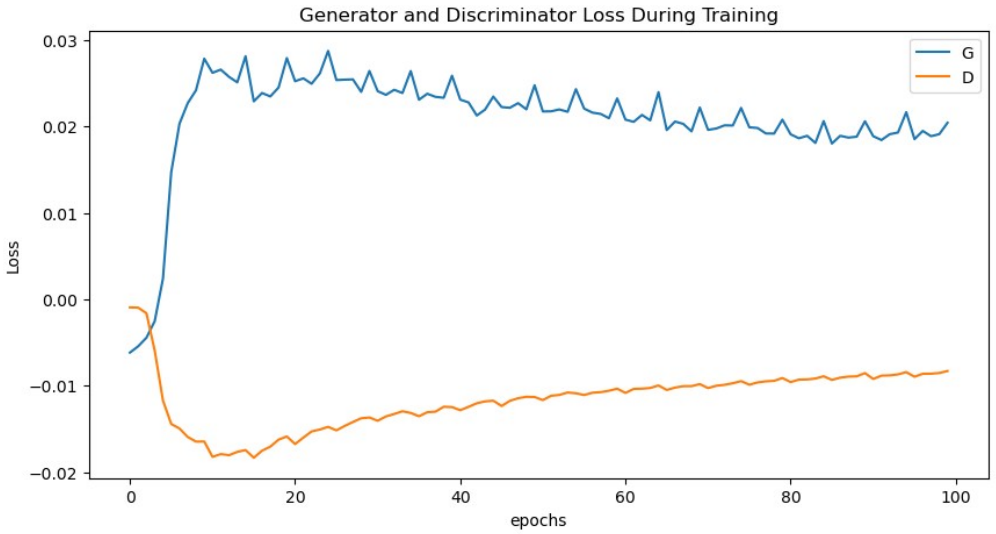


Figure 13. Loss of the generator and discriminator (wGAN)

The **wGAN-GP** model is trained for 200 epochs with 64 batch size. The loss of the generator and discriminator is shown in the following figure 14:

Figure 14. Loss of the generator and discriminator (wGAN-GP)

I also use the trained model from 200 epoch **wGAN-GP** to train another 50 epoch with 64 batch size. The loss of the generator and discriminator is shown in the following figure 15:

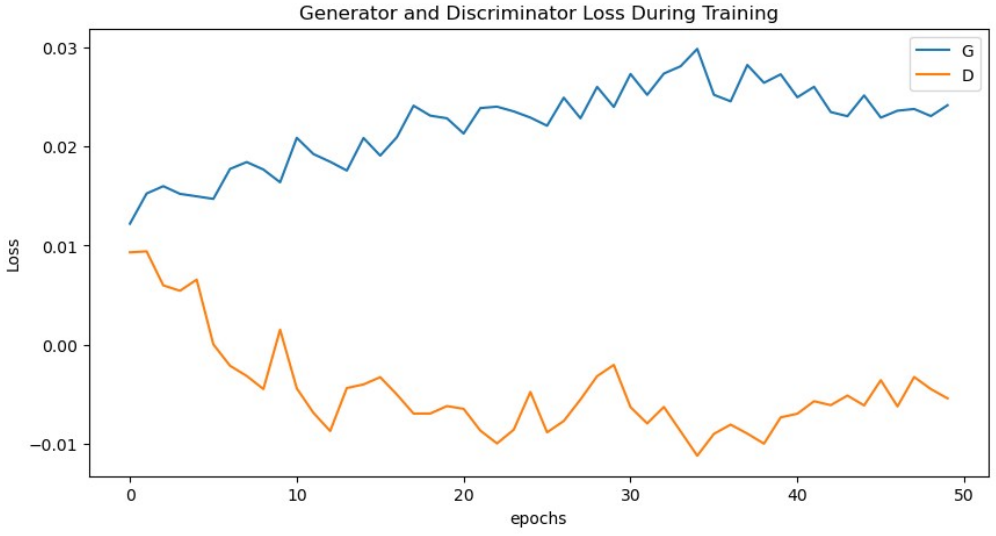


Figure 15. Loss of the generator and discriminator (wGAN-GP Retraining)

The data shown in these images clearly show that the loss function has a high stability and will only be fine-tuned within a small up and down range, which makes the generator effect more realistic.

4.3 *Generated Images*

The generated images of **wGAN** model is shown in the following figure 16:

Figure 16. Generated Images (wGAN)

The generated images of **wGAN-GP** and the retraining **wGAN-GP** model is shown in the following figure 17:

Figure 17. Generated Images (wGAN-GP)

However, the overall model of **wGAN-GP** is inferior to **wGAN**, the generated images of **wGAN-GP** are not as good as wGAN, especially when the **label is "Sandal"**.

The reason for this situation, I think is the model is not trained enough. The model is trained for 200 epochs, but the generate image is still bad. Therefore, I decided to retrain the model for another 50 epochs.

After retraining, the generated images of **wGAN-GP** are better than before, but still not as good as **wGAN**. Therefore, I decided to use the **wGAN** model to generate images.

5 *Generated Images with Different Labels*

There is a main function in the file loader which is use to generate images with different labels. The principle of function is loading the trained model and let user input the label. The model will generate 6 images with 32 x 32 pixel and the user also can choose which training model to use. The generated images are shown in the following figure 18:

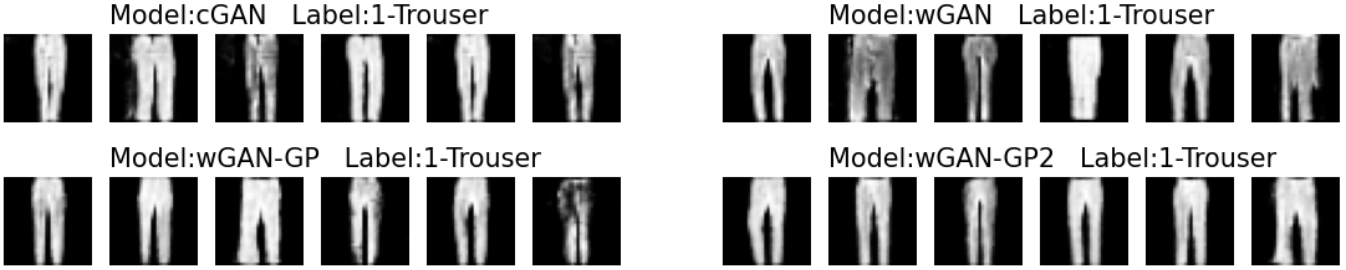


Figure 18. Generated Images with Different Labels (Label: 1-Trouser)
(wGAN-GP2 mean the retraining wGAN-GP model)

From the results of the four images, the generated images look realistic and work well, but not every label is like this. The generated images of label 5 (Sandal) is shown in the following figure 19:

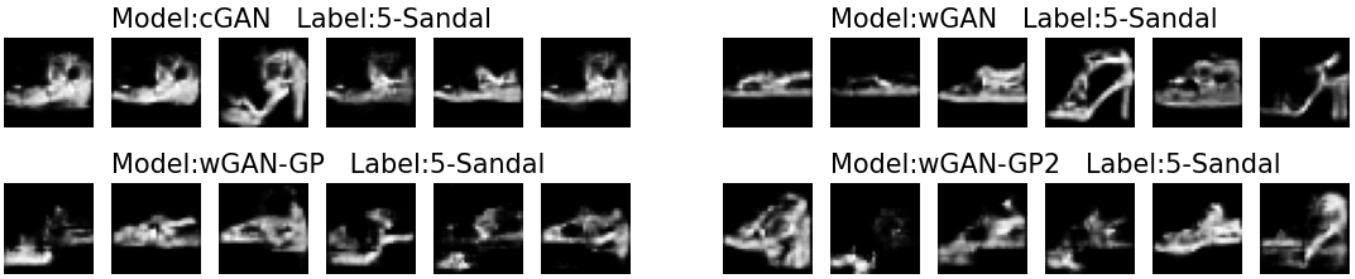


Figure 19. Generated Images with Different Labels (Label: 5-Sandal)
(wGAN-GP2 mean the retraining wGAN-GP model)

From the results of the four images, the generated images look realistic and work well only in **wGAN** model, but not in others.

6 *Future Research*

We try anothor dataset, which is the clothing dataset, to train the model. The dataset is from Kaggle. We use this dataset to test the generation of color images. The dataset is shown in the following figure 20. The dataset is divided into 10 labels, and visualization show in the figure 21.

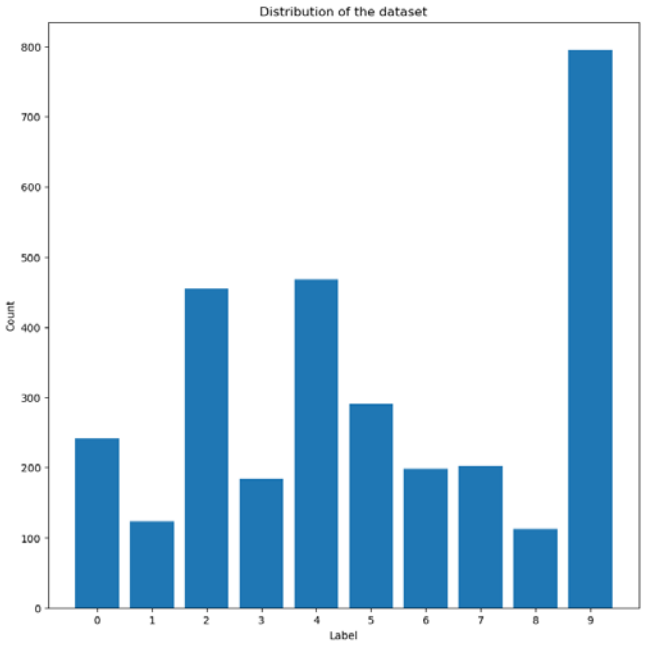
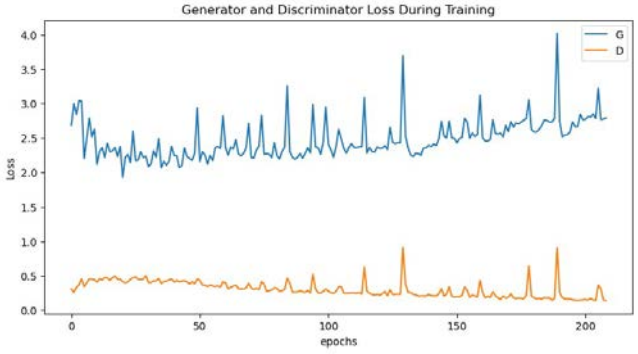


Figure 20. Clothing Dataset Label & Number

Figure 21. Clothing Dataset Visualization

We use the **cGAN** model to train the dataset, and the generated images and loss function are shown in the following figure 22:



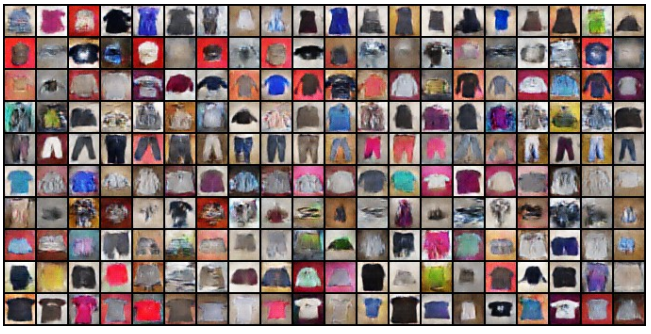


Figure 22. Clothing Dataset Generated Images and Loss Function

From the results of the generated images, the generated images look not very realistic. There are two reasons for this situation. The first reason is the train epoch is not enough as the generator and discriminator loss function almost no change. The second reason is convolutional layer is not enough because this dataset is color images, so the convolutional layer should be more than the fashion dataset.

7 Conclusion

In this project, our group has implemented a basic GAN model to generate images, and also optimized it into **cGAN**. In the subsequent optimization, I optimize the loss function direction of the group's **cGAN**, i.e., the **wGAN** and **wGAN-GP** models, to make the loss functions of the generator and discriminator more stable, which improves the realism of the generated images of the model. In comparing the four models (human judgment), uncovering the overall quality, **wGAN** is the best of the four model.

In the future, I will continue to optimize the model to make the generated images more realistic, which includes, retraining of the existing training model of **wGAN-GP** to arrive at a better fashion image generation, optimization and more iterations for the convolutional layer of **wGAN**. I think this project can also enhance some front-end development, like a web or application, which can make users have a better experience.

As a group project, I think the group members are very cooperative and the division of labor is clear. The group members are very responsible for their own work. In the future, I will continue to work with them to complete the project.

8 Reference

[1] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, “Are GANs Created Equal? A Large-Scale Study,” 2017, doi: 10.48550/arxiv.1711.10337.

[2] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” 2015, doi: 10.48550/arxiv.1511.06434.

[3] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” 2014, doi: 10.48550/arxiv.1411.1784.

[4] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, “Generative adversarial networks” 2014, arXiv:1406.2661

[5] P. Kancharla and S. S. Channappayya, "Improving the Visual Quality of Generative Adversarial Network (GAN)-Generated Images Using the Multi-Scale Structural Similarity Index," 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 2018, pp. 3908-3912, doi: 10.1109/ICIP.2018.8451296.

[6] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” 2017, doi: 10.48550/arxiv.1701.07875.