A Comparative Study of Generative Adversarial Networks for Clothing Image Synthesis

Name: Chen Ru Bing *SID: 21094526d*
Contact: ru-bing.chen@connect.polyu.hk

ABSTRACT

Generative Adversarial Networks (GANs) are powerful models for realistic image synthesis. However, applying GANs to clothing image synthesis poses several challenges, such as diversity, quality, and alignment of the generated images. In this paper, we present a comparative study of different types of GANs for clothing image synthesis, such as traditional GAN, conditional GAN (cGAN), Wasserstein GAN (wGAN), and others. We mainly use FashionMNIST dataset for training, and we also tried a colored-image dataset of clothes, which is "Full Clothing Dataset". The evaluation of the performance is figured out mainly by visual quality of the generated image. Overall, our model can generate fashion images with controllable labels and relative high quality images in terms of visual effects.

Keywords – Deep Learning, Computer Vision, Generative AI, Adversarial Networks, Image Generation

I. INTRODUCTION

With so many different and complicated types of clothes, fashion matching is an essential part of everyday life for modern people. Every day, people have to face the question: "What should I wear today?" Sheila, a professional fashion designer, is also having trouble coming up with new matching ideas. The goal of this report is to use generative AI models to help people like Sheila make better clothing choices by exploring the Generative Adversarial Networks (GAN).

Generative Adversarial Networks (GANs) are a class of neural network models that can learn to generate realistic and diverse images from random noise. GANs consist of two components: a generator and a discriminator (Fig.1). The generator tries to fool the discriminator by producing fake images that resemble the real ones, while the discriminator tries to distinguish between real and fake images. The two components are trained in an adversarial manner, where the generator aims to maximize the probability of the discriminator being wrong, and the discriminator aims to minimize it.

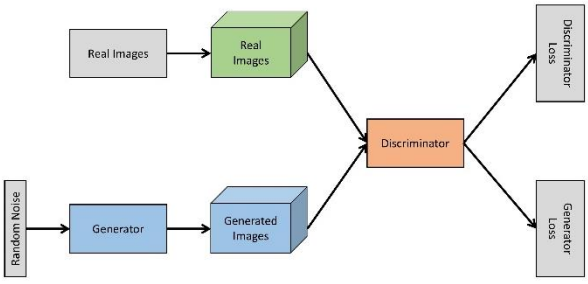


Figure 1 – Structure of GAN Model (https://medium.com/analytics-vidhya/coding-your-first-gan-algorithm-with-keras-ab2bdf761746)

The goal of GANs is to find a balance where the generator produces realistic images and the discriminator cannot tell them apart from the real ones (shown below [1]).

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

To achieve this goal, GANs use loss functions that measure how well the generator and the discriminator are performing. The loss functions are usually based on some distance metric between the distribution of the real data and the distribution of the fake data generated by the generator as shown above. The generator tries to minimize this loss function, while the discriminator tries to maximize it. This creates a competition between the two networks, where the generator tries to fool the discriminator and the discriminator tries to catch the generator.

However, traditional GAN model have several shortcomings, including the training instability. GANs can be difficult to train, with the risk of instability, mode collapse, or failure to converge. This is because the generator and the discriminator are competing against each other, and their objectives are not aligned. If one network becomes too strong or too weak, the other network may not learn anything useful. Furthermore, the gradient of the two loss functions have the possibility to be too small or too large, which may lead to the vanishing or exploding of the gradients. Also, GANs do not have a clear and objective way to measure their performance and quality. This is because GANs do not have an explicit likelihood function or a predefined metric to compare the generated data with the real data.

In order to make an improvement to the traditional GAN, we tried conditional GAN (cGAN) and Wasserstein GAN (wGAN) to further explore the performance of generated image with control labels.

II. EXPERIMENT

2.1 Dataset

In this report, we tried 2 datasets on fashion and clothing, which are FashionMNIST and Full Clothing.

FashionMNIST dataset is a collection of 70,000 grayscale images of 10 types of clothing items, such as shirts, dresses, shoes, etc. The images have a size of 28x28 pixels and are labeled with an integer from 0 to 9. The dataset is divided into a training set of 60,000 images and a test set of 10,000 images (visualization shown in Fig 2).

Figure 2 – Visualization of FashionMNIST Dataset with all labels having 10 images as the samples

Another dataset is for improving the generalization of model by training it on coloring images which has 3 RGB channels. This dataset contains 10 labels which are different from the FashionMNIST but consists popular categories in clothing among daily life. The dataset is a subset of the full clothing dataset (https://github.com/alexeygrigorev/clothing-dataset) with the top-10 most popular classes. (Shown in fig 3, 4)

Figure 3 – Visualization of Full Clothing Dataset

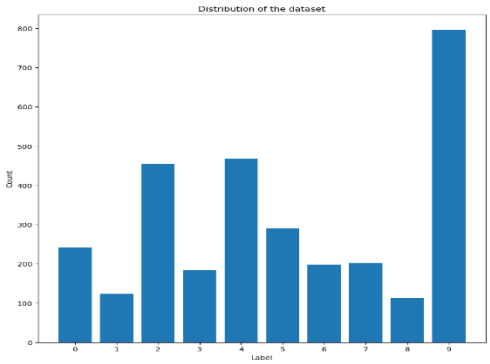


Figure 4 – Distribution of Full Clothing Dataset

2.2 Environment Setup

The model is constructed by python and mainly rely on the pytorch framework. Environmental setting and packages used are shown in Table1.

Package	Version
python	3.9.12
matplotlib	3.6.2
opencv	4.6.0
Numpy	1.22.3
Pytorch-cuda	11.6

Table 1 Environment Settings

III. TRADITIONAL GAN

3.1 Capability

In our first trial, we used the template of GAN model consisting of generator and discriminator models inside it. The models are basically the multi-layer perceptron with all the layers are ***linear*** transformation. We generate 5 images each time as the fake images and make them as the dataset for discriminator to recognize. The loss function is Binary Cross-Entropy Loss, which is a common loss function for binary classification problems. It measures the difference between two probability distributions: the true labels and the predicted probabilities.

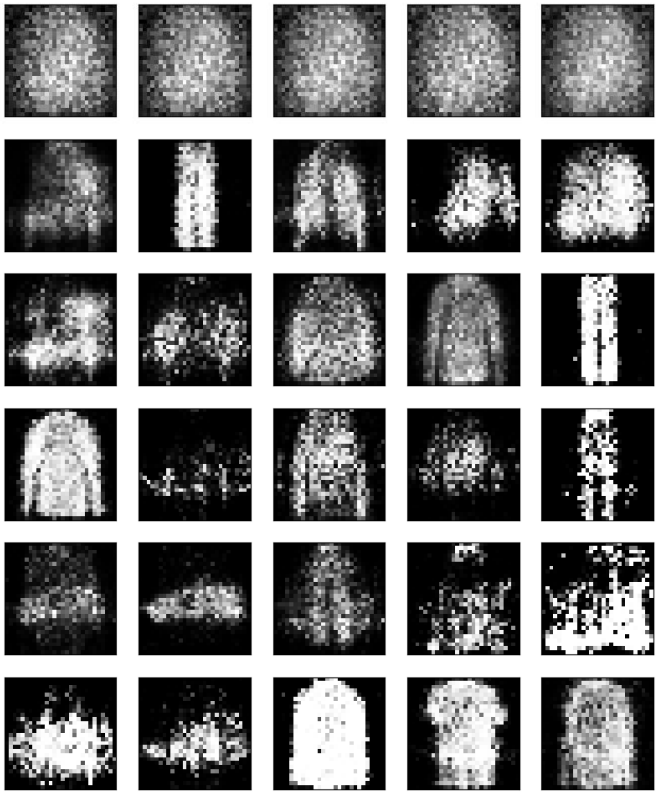


Figure 5 – Generated images for epoch (01, 20, 40, 60, 80, 100)

We can figure out the process of training inside the GAN model from the fig.5, that the generator initially synthesis 5 images of random noise with given random seed (hyperparameter to be modified). After that, each iteration of the generator will perform better and better, leading to the blurred outlines that can slightly moving towards the original dataset. The convergence of loss functions of both generator and discriminator is shown in Fig 6 for better interpretation.

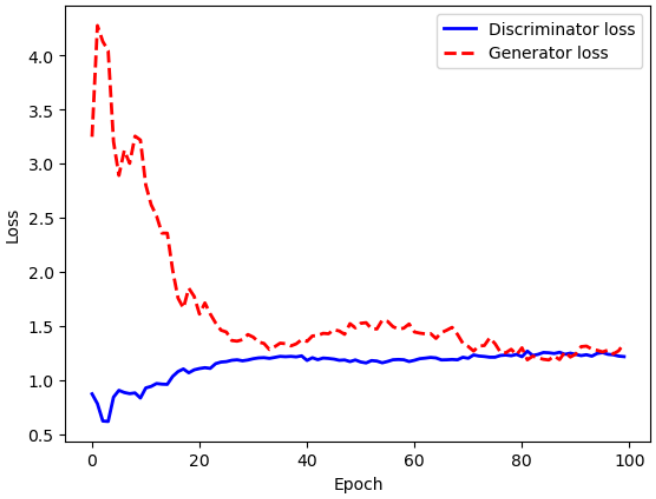


Figure 6 – Loss Convergence of Traditional GAN Model

We can conclude from the diagram, that the loss functions are getting closer to achieve the balance, which is exactly what we need for the GAN model's training. Generator (G) achieves a super high loss comparing to the Discriminator (D), indicating that the loss of purely ransom noise is indeed high because there is no data being fed so far. D performs well initially in the task of recognizing the real and fake images, however it becomes worse when facing the improvement of G model, until it cannot recognize the reality of the input image, and completely confused with the images being generated with the original dataset.

3.2 Shortcomings

3.2.1 Random Label

As we shown in the figure 5, the generated images are random without any specifying of the labels. This can satisfy the partial goals of our task but still not enough for actually implementation in the real life situation. Ideally, we would like to control the label of generating, that the model should be able to generate different styles of trousers when given the "trouser" label.

This can be improved by adding conditions when training the generator and discriminator, involving the consideration of data's labels from dataset. The improvement is achieved in the further development.

3.2.2 Bad Image Quality

In the output images in traditional GAN model, we can figure out that the image quality is still unsatisfied, which consisting not only the problem of low image resolution, but also blurring outlines, indistinct details, unclear features, and overload noises.

One approach is to improve the image quality of output images, that we can denoise the image by applying Gaussian algorithm or improve the image resolution by adding more pixels onto output image sizes. However, they are not touching on the core drawbacks of the traditional implementation.

The major reason for bad image quality is the simple linear dense functions when forwarding the fully connected layers, which may lead to fast but poor abstraction of the original datasets. Instead, we can apply a Convolutional Neural Network to further extract the features of the dataset, therefore the image generated can learn from the features, but not only from the arrange of the pixels. Thus, better generation quality can be achieved. This is also be done in the future development.

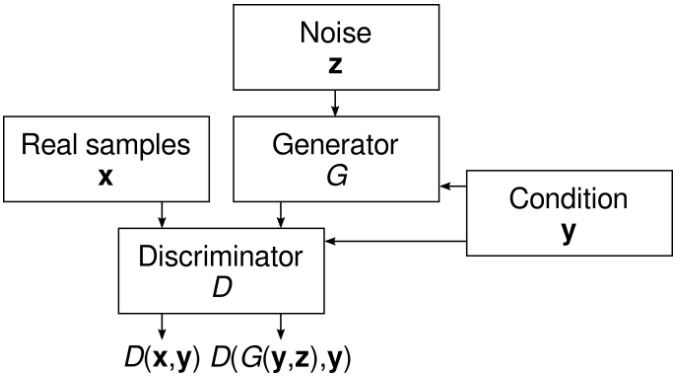
3.2.3 Poor Variation

Due to the simple network structure being used in the traditional GAN model, the generated images are lack of variation, that the images are seriously lost the details and the also the styles of the original clothes. By adding more convolutional layers to extract the features from dataset, we can improve this shortcoming in the future development as well.

IV. IMPROVEMENT FROM GAN

4.1 Conditional GAN and Convolutional GAN

According to the shortcomings we've analyzed from the previous sections, thus we prompt the following improvements: adding conditional labels and CNN to control the output label and extract the features of images. This is done by conditional GAN and convolutional GAN respectively.



A Conditional Generative Adversarial Network (cGAN) is a variant of GAN that generates images based on some additional information, such as a class label, a text description, or another image. A cGAN consists of two neural networks as well: a generator and a discriminator. The generator creates fake images that resemble the real ones, while the discriminator tries to tell them apart. Unlike a standard GAN, both the generator and the discriminator take the additional information as an input, along with the image or the noise vector. This allows the generator to produce images that are not only realistic but also consistent with the additional information. The discriminator can also leverage the additional information to make more precise judgments.

Figure 7 – Output of cGAN when epoch = 60

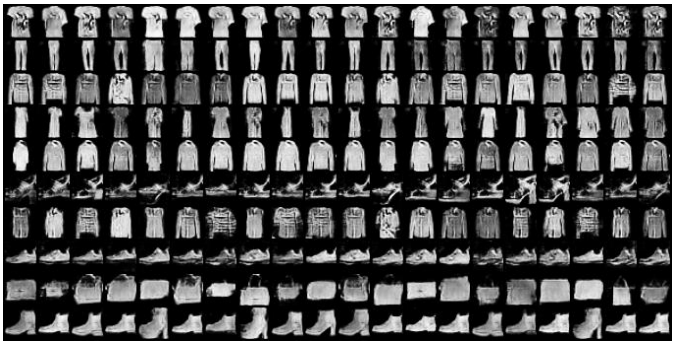


Figure 8 – Output of cGAN when epoch = 100

Among the 100 epoches we've run during the training, we conclude from human's subjective judgement that the epoch 60 performs best comparing to the other trails. We can figure out from the Figure 7, that in epoch 60's experiment, the generated images are well labelled that each row contains a category of the clothes. The images in one label can be distinguished well. For example, in figure 9, we analyze that even though they are in the same label (e.g. dress), we can clarify the difference between the individual objects, such as long sleeves and short sleeves of dresses, strap and without strap in the bag category.

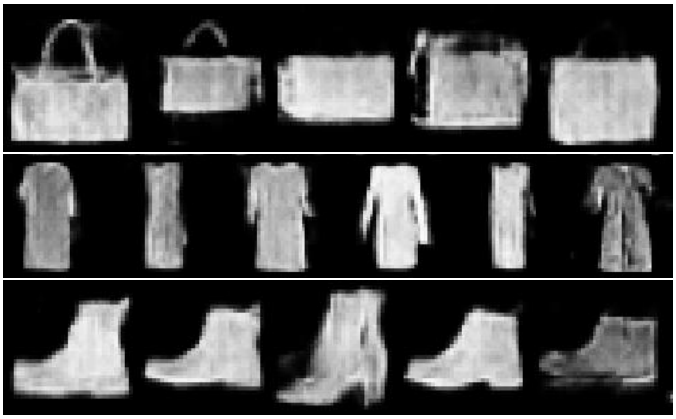


Figure 9 – Variation of Clothes under one category

In order to achieve this, we also applied CNN onto the dataset to extract the features from the clothes instead of just embedding pixels as the traditional methods. We used 4 convolutional layers to figure our the general distribution of the dataset and thus can generate better quality images from the generator, which is fully optimized based on the features studying from the CNN.

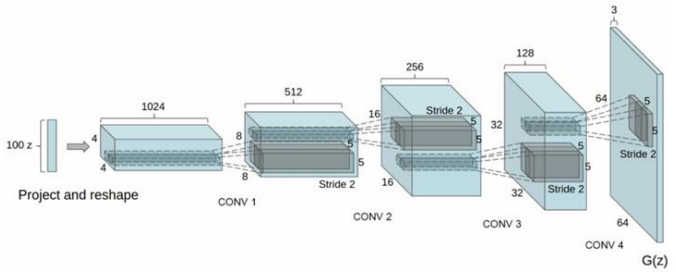


Figure 10 – Convolutional Layers in Model

4.1.1 Shortcoming

However, when we analyze the figure 8, containing the generated images from epoch 100, we can see clear noises and unexpected texture in the clothes (Fig 11 as example). There are somehow twisted textures inside the T-shirt and also other images.

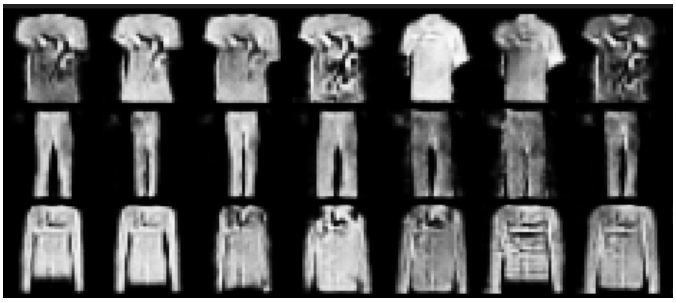


Figure 11 – Unexpected Textures

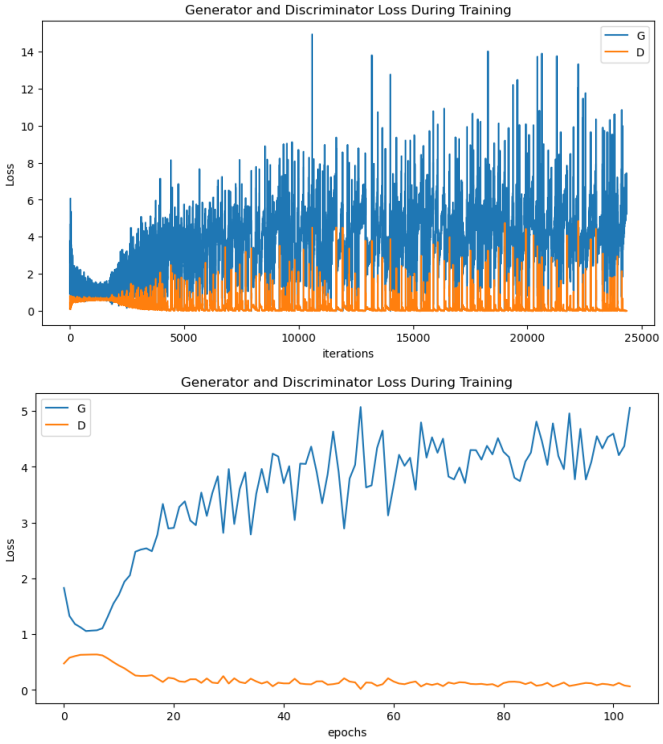


Figure 12 – Loss Convergence of cGAN

We are still trying to figure out the reason for those unexpected features inside images when epoch is greater than 60. We conclude it as somehow the overfitting, but still cannot concisely give a complete interpretation, since we cannot extract any unusual from the loss convergence diagram from Fig 12. The loss's trend is super instable that they are progressing like the zigzag lines. Therefore, improving the stability can be one way to breakthrough.

Therefore, we are trying to improve the model by a more stable model with better performance on the image quality of generation. This is done by implementing wGAN in additional to the cGAN, which will be shown in the further development.

4.2 Wasserstein GAN

wGAN stands for Wasserstein Generative Adversarial Network, which is a variant of GAN that uses a different loss function and a different way of updating the discriminator and the generator. It can also be applied to conditional image generation, where the generator and the discriminator receive some additional information as an input. Comparing to the cGAN, wGAN improves in several aspects:

Stability: wGAN avoids some of the common problems of GAN training, such as mode collapse, vanishing gradients, or

failure to converge. This is because wGAN uses a different distance metric and loss function that have better theoretical and empirical properties. The distance metric is the Wasserstein distance, which measures how much mass needs to be moved to transform one distribution into another. The loss function is the Wasserstein loss, which is proportional to the Wasserstein distance. Unlike the standard GAN loss function, which can be very noisy and unstable, the Wasserstein loss function provides a smooth and meaningful gradient signal for both the discriminator and the generator.

Performance: wGAN generates higher quality and more diverse images than cGAN. This is because wGAN better captures the distribution of the real data and avoids mode collapse, where the generator produces similar or identical images for some categories. Moreover, wGAN provides a more reliable way to measure the performance and quality of the GAN model. The Wasserstein loss function can be used as a proxy for the image quality, as it correlates well with the visual realism and diversity of the generated images.

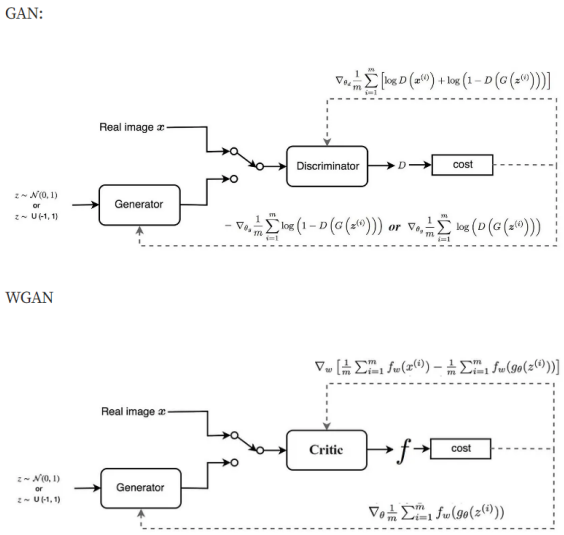


Figure 13 – Structure of GAN and wGAN (https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490)

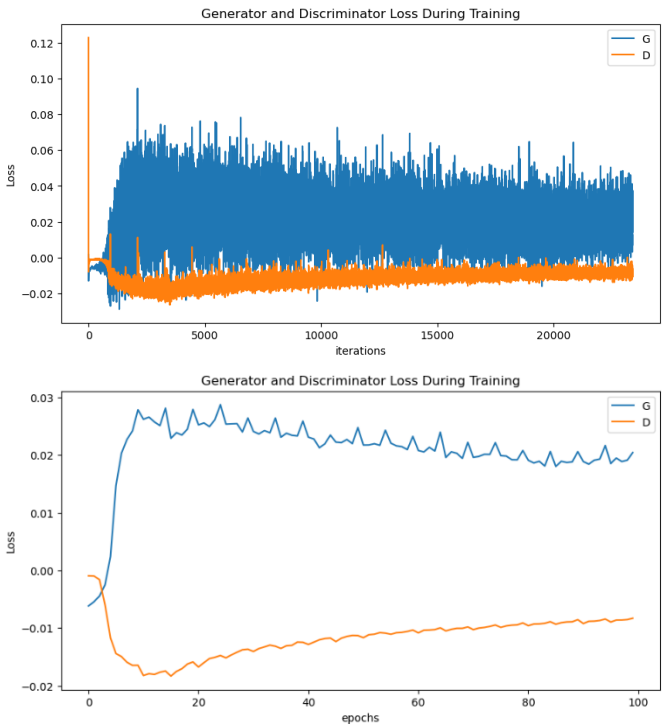


Figure 14 – Loss Diagram of wGAN

We modified the cGAN to wGAN, and the performance indeed improves as shown in figure 14. The loss diagram shows that the convergence is much more stable than the cGAN approach above, and the gap between G and D are decreasing as the epoch increasing, which is exactly fitting our goal of training.

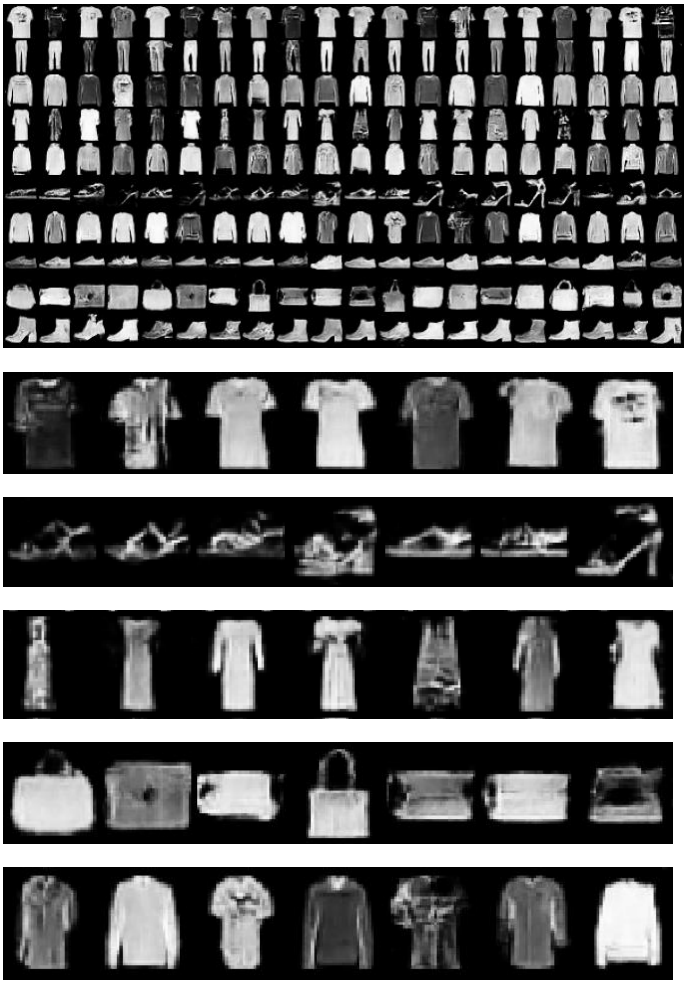


Figure 15 – Generated Images from wGAN

As shown in the figure 15, the output images are more stable and more various comparing to the cGAN, that we can generate different types of clothes in one category. Also, the quality of the image is improved significantly as well, since there is nearly no noise spreading around the clothes and even the different patterns are generated concisely.

4.2 Wasserstein GAN – Gradient Penalty

In order to further improve the performance of the model, we also tried Wgan-GP to have a taste of the overall improvement. wGAN-GP stands for Wasserstein Generative Adversarial Network with Gradient Penalty, which is a variant of GAN that uses a different regularization method for the discriminator.

wGAN-GP enforces a Lipschitz constraint on the discriminator by adding a gradient penalty term to the loss function. This term penalizes the discriminator if the norm of its gradient deviates from 1. This avoids the problem of weight clipping in wGAN, which can cause undesired effects such as exploding or vanishing gradients, or poor approximation of the Wasserstein distance. The gradient penalty term ensures that the discriminator has a smooth and bounded gradient everywhere,

which improves the stability and performance of the GAN model.

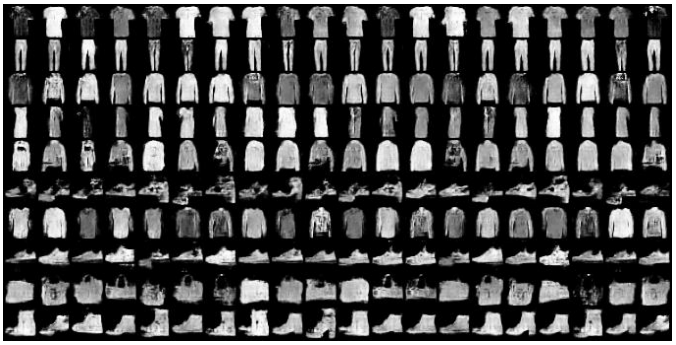


Figure 16 – Output of Wgan-GP

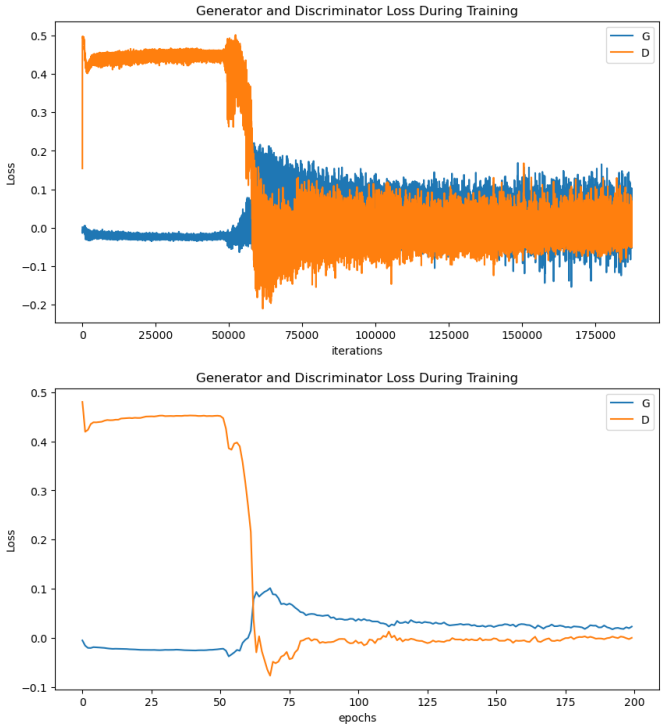
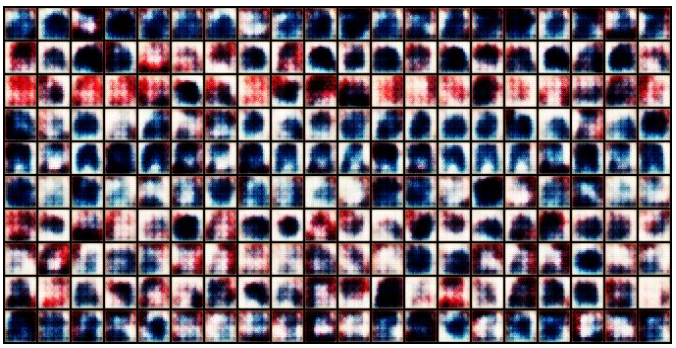


Figure 17 – Loss Diagram of wGAN-GP

V. FURTHER IMPROVEMENT

As mentioned in the dataset introduction, we not only train the FashionMNIST dataset, but also tried a colored images with 10 different labels of clothes.

We apply wGAN, which is the best performed model for training the colored dataset. The images generated from the model among different epochs are shown below.



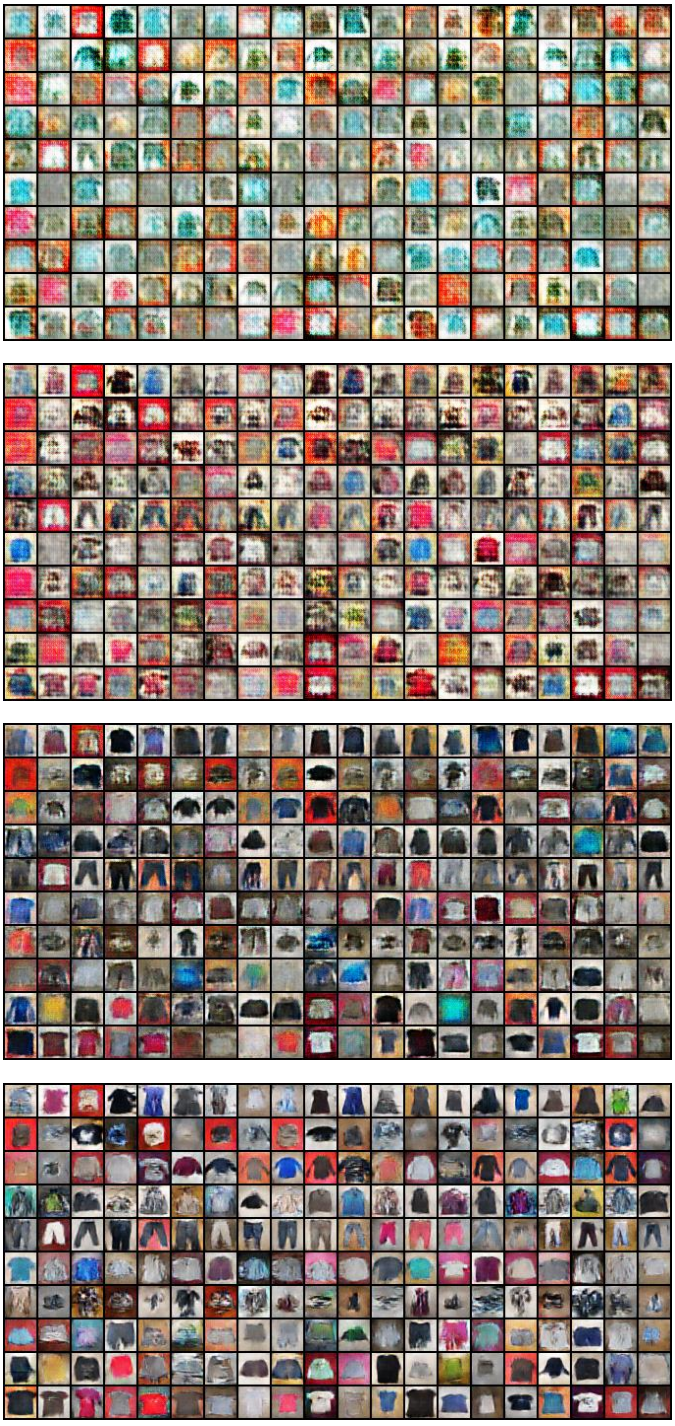


Figure 18 – Output of Color Image Generation of epoch (1, 50, 100, 150, 200)

In order to save time and resources, we resize the original dataset of Full Clothing to the image size of 28*28, which is the same as MNIST to achieve faster processing. We can analyze from the outputs, that the details, patterns and colors of the images in one category can be preserved well and we can obtain more details due to the addition of color dimensions.

However, due to the limitation of the resolution of the image, we still cannot achieve a satisfying result. But it is feasible to implement more CNN to resolve the problem and gain better performance in the future development.

For the further improvement, we can also exploring different distance metrics and loss functions for GANs, such as the f-divergence, the maximum mean discrepancy, or the sliced Wasserstein distance. These metrics and functions may have different properties and advantages for measuring the similarity between distributions and optimizing the GAN model.

VI. SUMMARY

In this report, we compared the performance of different GAN models such as traditional GAN, conditional GAN, convolutional GAN, wGAN, and wGAN-GP. By implementing the different approaches and improving the model time by time, we gained valuable knowledge in progressing and tuning the model to achieve our goal in a more efficient and stable way of performance. We conclude that among all the approaches so far, the wGAN has the best performance according to the evaluation from subjective human's perception. However, we recommend user to try different models in the program we've provided to have a taste of all the effects.

REFERENCE

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative Adversarial Networks. Communications of the ACM, 63(11), 139–144. https://doi.org/10.1145/3422622