**A Comparative Study of Generative Adversarial Networks**

**for Clothing Image Synthesis**

*Name: Chen Ru Bing SID: 21094526d*

Contact: [ru-bing.chen@connect.polyu.hk](mailto:ru-bing.chen@connect.polyu.hk)

**ABSTRACT**

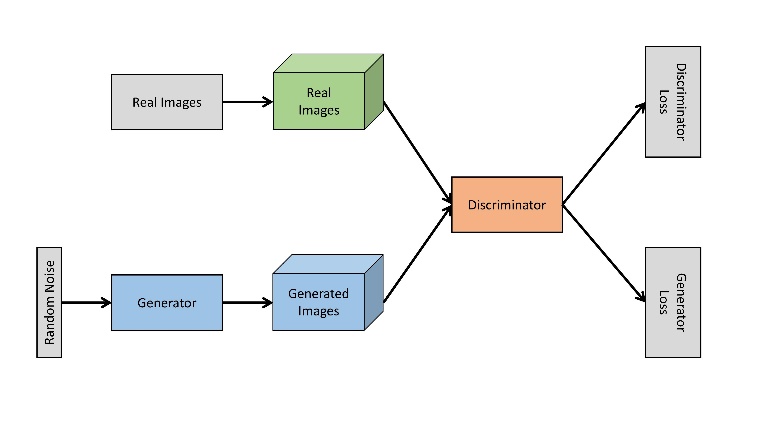
Generative Adversarial Networks (GANs) are powerful models for realistic image synthesis. However, applying GANs to clothing image synthesis poses several challenges, such as diversity, quality, and alignment of the generated images. In this paper, we present a comparative study of different types of 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**

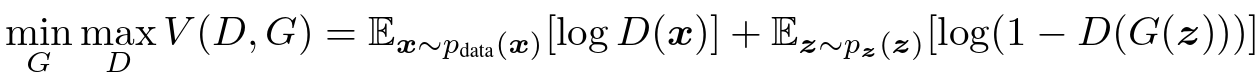
With so many different and complicated types of clothes, fashion matching is an essential part of everyday life for modern people. Every day, people have to face the question: “What should I wear today?” Sheila, a professional fashion designer, is also having trouble coming up with new matching ideas. 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]).



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).

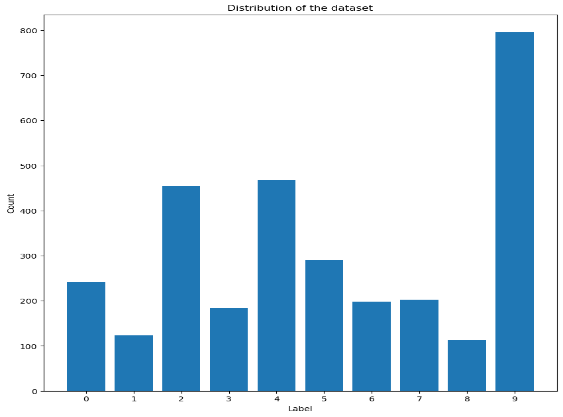


**Figure 2** – Visualization of FashionMNIST Dataset with all labels having 10 images as the samples

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.

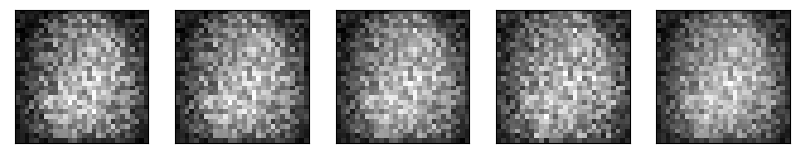
|  |  |
| --- | --- |
| Package | Version |
| python | 3.9.12 |
| matplotlib | 3.6.2 |
| opencv | 4.6.0 |
| Numpy  Pytorch-cuda | 1.22.3  11.6 |

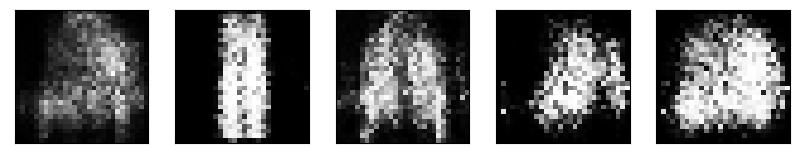
**Table** 1 Environment Settings

**III. TRADITIONAL GAN**

***3.1 Capability***

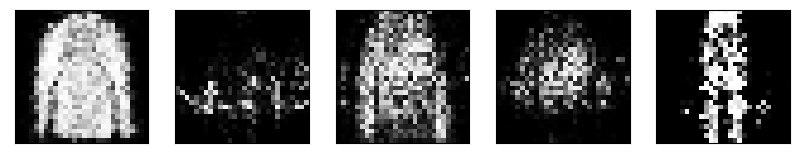
In our first trial, we used the template of GAN model consisting of generator and discriminator models inside it. The models are basically the multi-layer perceptron with all the layers are ***linear*** transformation. We generate 5 images each time as the fake images and make them as the dataset for 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.





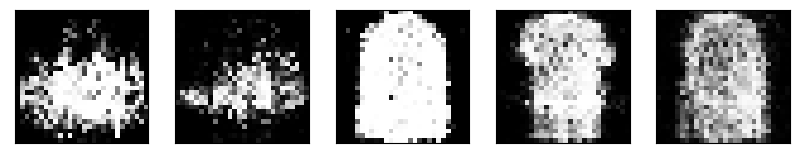
A picture containing text, different

Description automatically generated



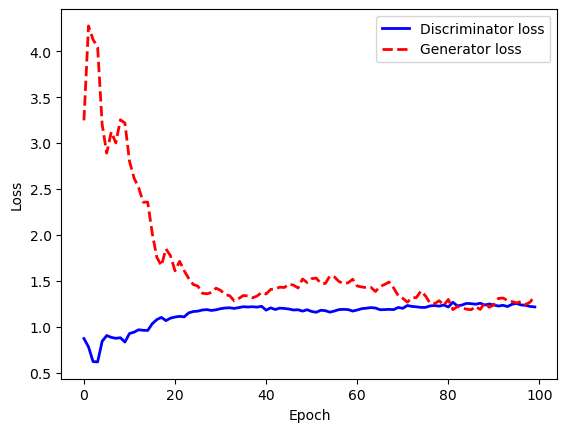
A picture containing text

Description automatically generated



**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.

IV. IMPROVEMENT FROM GAN

4.1 Conditional GAN

REFERENCE

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