Generating Full-Body Standing Figures of Anime Characters and Its Style Transfer by GAN

A Thesis Submitted to the Department of Computer Science and Communications Engineering,
the Graduate School of Fundamental Science and Engineering of Waseda University
in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: July 18th, 2020

Zepeng Liu
(5118FG43-1)
Advisor: Prof. Hiroshi Watanabe
Research guidance: Research on Audiovisual Information Processing

Abstract

We discussed the anime characters' full body illustrations generation methods based on generative adversarial networks (GANs). This research is expected to be applied in the field of animation and game, synthesized images that can be used as game material to significantly reduce production costs or provide new design inspiration for professional character designers.

To smoothly create the generative model, firstly, we make a new dataset. This dataset contains 12,000 high-quality images of game characters, providing a guarantee for subsequent experiments. Secondly, we investigate several GANs and selected StyleGAN as our experimental benchmark. We train 3 models of StyleGAN. Among them, the best model can generate high quality standing pictures with a resolution of 512×512 and an FID of 5.02.

By further analyzing the latent space of our generated model, we find that StyleGAN can effectively separate the characteristics of anime characters. It can perform feature mixing and feature transformation through latent code interpolation. This property provides a lot of interesting possibilities, such as keeping the character's clothing unchanged, and controlling character's actions to make simple animations.

Key words: GANs, StyleGAN, image generation, style transform, high-resolution, latent code

Acknowledgments

First of all, my deepest acknowledgments should go to Prof. Hiroshi Watanabe, who give me the most enthusiastic guidance and the best hardware support. He was the enlightener and protector of my research career in Japan.

Then, I should express my heartfelt gratitude to every members in Watanabe Lab, who have accompanied me through the 2 years at Waseda University, sharing happiness and sadness with me.

Finally, my thanks would go to my beloved family for their constant overall support and loving considerations through my all life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With the rise of the third artificial intelligence boom, artificial intelligence applications based on machine learning and neural networks have demonstrated their strong influence in various fields. Especially in the field of computer vision, face recognition, target detection, and image segmentation algorithms based on machine learning have been quite improved and gradually tend to human level. [19, 23–25] Now, they have shown their remarkable economic value in all walks of life.

On the other hand, researchs on image generation is much more difficult. They are less successfully used in business right now. Image generation models use neural network to learn the image distribution in huge data, ultimately synthesize images similar to the original data. In machine learning, image generation is a regression task, which is much more difficult than a discrimination task, because the generative model must output richer information based on a smaller input.

Image generation models can be roughly divided into two categories, unsupervised models and supervised models.

The unsupervised image generation model is more general, it only cares whether the generated image is similar to samples in the dataset, and does not care about its classification, so it is difficult to control. An unsupervised image generation model likes a Gaussian pseudo-random number generator. Although a specific output cor-

responds to a determined input random seed, we do not know the physical meaning of the random seed. We can not effectively control the output of the model; we can only choose the pictures we want from a large number of messy results. It can be said that the unsupervised image generation model is a crazy artist with unlimited creative power but arrogant. Generally, applications of unsupervised models are relatively limited, considered for artistic creation or data enhancement. Although there was an artificial intelligence work called *Edmond de Belamy* that was auctioned at a striking price of $432,500, most generative models are still synthesizing faces or other simple objectives, and rarely create truly valuable art works.

In contrast, supervised models are easier to use. We add various conditions when training the model to ensure that it can achieve our requirements in practical utilization. Unlike training unsupervised models, training supervised models requires labeling datasets, which can be text information or paired pictures. For example, we can use a paired line draw and original painting to train an automatic colorization model. Supervised models are more widely used, such as segmentation, mapping, image repair, super-resolution, Style Transform, etc.

Generative adversarial networks (GANs) [4] are now the mainstream technology in image generation technology. Since GANs were proposed by Ian Goodfellow in 2014, it has received more and more attention from academia and industry. With the rapid development of GAN in theory and model, it has more and more in-depth applications in computer vision, natural language processing, human-computer interaction, and continues to extend to other fields. In recent years, GAN has made great progress in image generation, and some advanced models have been able to generate realistic high-resolution images.

Traditional 2D games or animation productions in Japan require a lot of manual labor and material resources. Well-trained character designers often require years of professional education. Even so, a successful character design often takes weeks or even months. The market price of a professional character design is often more than 100,000 yen.

A finished product of character design is generally a drawing of a character in a

standing pose without background (hereinafter called the *standing painting*). Using GANs to generate character full body standing painting is a very potential research. The generated standing paintings can be directly used as materials in games, reducing the production cost of animations or games. Moreover, one game maker can create a novel game even if he does not know art knowledges. On the other hand, creators may be able to get inspirations from the huge generated images to create new illustrations. Beginners in character design can also simply obtain training materials.

1.2 Research Objectives

Now, GANs [4] has achieved success in many fields, many images generated by GANs are hard to distinguish by human eyes. But it still failed to create standing painting.

Most image generation models are made for human face generation. Human face generation is a relatively simple task. Representative open-source human face datasets include CelebA [28], CelebA-HQ [13] and FFHQ [14]. They all strictly limit the position and orientation of the face, even the positions of the eyes, nose and mouth are mostly fixed. It is not difficult to learn by an advanced neural network. On the contrary, the mainstream illustration dataset is Danbooru2017. The pictures' quality in the dataset is very unstable, which contain professional illustrations and novice line drafts. Background and number of characters in one picture is also uncertain. Moreover, characters in one standing painting are rich in movements, the heights and actions of characters are changeful, the proportion also variable, which bring great difficulty to make a new dataset.

Human face pictures are recognizable from the resolution of 64×64. However, standing paintings need larger area to display character's whole body, which's resolution less than 256×256 are difficult to identify. Traditional generative models need to face many problems such as unstabled training and mode collapse. They merely synthesize images above 200×200. Advanced high-resolution generation models require huge calculation time and memory.

This article is dedicated to using advanced GANs to generate high-quality full-body standing painting, and explore the application of synthesized standing. Therefore, we have the following 4 objectives.

1. Verify that GANs can effectivelyh synthesize illustrations and find a available matric to evaluate the quality of illustrations.
2. In order to graduate the feasibility of subsequent experiments, we will create a new standing painting dataset.
3. Evaluate mainstream models to find the most suitable method for this research.
4. Analyze the experimental results and explore new applications.

1.3 Outline

Chapter 1: The Introduction of this paper. In this chapter, the background and objectives will be introduced.

Chapter 2: Previous works of this paper. In this chapter, several classical GANs will be introduced. It also introduced 2 kinds of typical metrics.

Chapter 3: The preparative experiment. We ensured that GANs can effectively synthesize illustrations with an self-built dataset of anime character' faces. Moreover, we verify the feasibility of FID in evaluating illustrations.

Chapter 4: The main chapter of this paper, try to synthesize full body standing painting by a advanced generative model. Moreover, explore the application of standing painting generation.

Chapter 5: Conclusion and Future Work is shown in this chapter.

Appendix: The Appendix of this paper. Some extra pictures and codes will be shared in this chapter.

Chapter 2

Previous Work

2.1 GANs

GANs, this new member of machine learning is already flourishing. As of June this year, there are currently at least 400 papers or variants of GANs.

Researchs on GANs can be divided into two lines. One is the theoretical line. It studies mathematically how to solve the instability and mode collapse of GANs, or re-explains it from different angles such as information theory and energy-based models. The second is the application line, dedicated to applying GAN to the field of computer vision, using GAN for image generation (e.g. specified image synthesis, text to image, image to image, video) and applying GAN to NLP or other fields. [9, 16, 33, 34] The use of GAN for image generation and conversion is currently the most studied, and research in this field has proved the great potential in image synthesis.

2.1.1 Vanilla GAN

Vanilla GAN proposed a new framework for estimating generative models via an adversarial process. [4] It used two networks, a generative network called generator (G) and a discriminative network called discriminator (D). The G used to capture date distribution, this network can synthesize data from a given noise (can be considered as random seeds). The D used to estimate whether the sample from the graining data

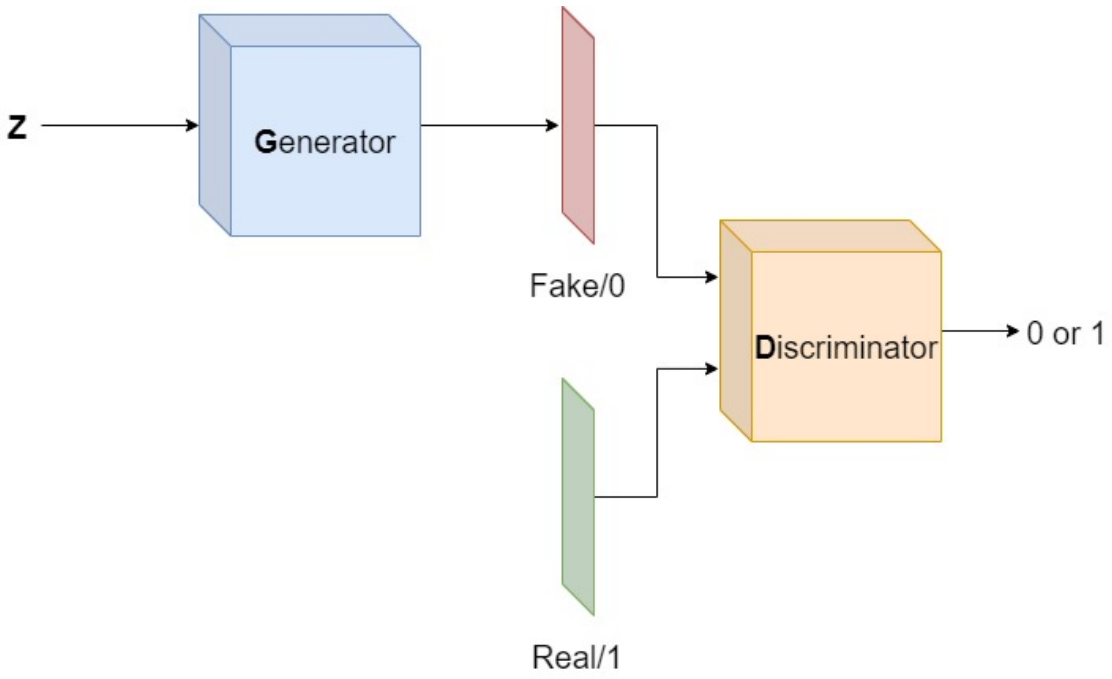

Figure 2-1: General Structure of GANs

rather G.

The loss function of vanilla GAN is Equation 2.1 and Equation 2.2. x means real images, z is habitually called as latent code (generally a uniform noise or a normal noise). The $G(z)$ means generated images.

The discriminative loss tries to reduce itself by increasing $D(x)$ and decreasing $D(G(z))$. The generative noise is just the opposite. We can think that D provides G with a dynamic loss function.

$$L_D^{GAN} = E[log(D(x))] + E[log(1 - D(G(z)))] \tag{2.1}$$

$$L_G^{GAN} = E[log(D(G(z)))] \tag{2.2}$$

The two networks are advancing in the competition. The training continues, and the data obtained by the generating network becomes more and more perfect. This approach is a pioneering work to improve the performance of generative model.

2.1.2 DCGAN

The main contribution of the paper is to provide a good network topology for GAN training. DCGAN is a big improvement after vanilla GAN, it bridged the gap between CNN (Convolutional Neural Networks) and GANs. [18, 22] The main improvement is in the network structure. It proposed and evaluated a set of constrains on the architectural topology of convolutional GANs. This work clearly improved the performance of the vanilla GAN and made it easier to train.

DCGAN's network structure is shown as Figure 3-1. Compared with the vanilla GAN, DCGAN almost completely uses convolutional layers instead connected layers. And the discriminator's structure is almost symmetrical to the generator. The entire network does not have a pooling layer. In fact, it uses deconvolution (fractionally-strided convolution) layers instead of upsampling layers to increase the stability of training.

So far, the network structure of DCGAN is still widely used. DCGAN greatly improves the stability of GAN training and the quality of generated results.

2.1.3 WGAN

WGAN proposed a new training method and proved it can converge finally. [1] That is mean training WGANs does not require maintaining a careful balance in training of the discriminator and the generator; and does not require a careful design of the network architecture either.

Unlike DCGAN, WGAN mainly improves vanilla GAN from refining its loss function. Even on the full connected layer, WGAN's method can obtain better performance results. WGAN's improvements to GAN mainly include 2 points. Firstly, removing the last sigmoid layer of discriminator. Then, The loss of the generator and discriminator does not need to be logarithmic. The loss function of vanilla GAN is Equation 2.3 and Equation 2.4.

$$L_D^{WGAN} = E[D(x)] - E[D(G(z)] \tag{2.3}$$

$$L_G^{WGAN} = E[D(G(z))] \tag{2.4}$$

WGAN' implementation seems to be quite easy, but it is very meaningful. WGAN theoretically gives the reason for the instability of GAN training, that is, cross entropy (Jensen-Shannon divergence) is not suitable for measuring the distance between uncorrelated distributions. To solve this problem, it uses Wasserstein distance to measure the difference between generated data distribution and the ground truth distribution. This approach theoretically solves the problem of unstable training and collapse mode. Leading to more diverse generated result.

2.1.4 WGAN-GP

WGAN-GP fixed some potential problem of WGAN, improved continuity limit conditions. [5] A new Lipchitz continuity limitation method gradient-penalty is proposed to solve the problem of gradient explosion and gradient disappears. Compared with the standard WGAN, WGAN-GP enables stable training of a wide variety of GAN architectures with almost no hyperparameter tuning. It has faster convergence speed than standard WGAN and can generate higher quality samples. The loss function of WGAN-GP is shown by Equation 2.5, Equation 2.6 and Equation 2.7

$$L_D^{WGAN\_GP} = E[D(x)] - E[D(G(z)] + GP \tag{2.5}$$

$$L_G^{WGAN\_GP} = E[D(G(z))] \tag{2.6}$$

$$GP = \lambda E[(|\nabla D(\alpha x - (1 - \alpha G(z)))| - 1)^2] \tag{2.7}$$

2.1.5 MakeGirlsMoe

MakeGirlsMoe is a successful application by GANs to generate face icon of Japanese anime character. This article combines several advanced methods. They used a DRAGAN [17] based ACGAN [21] which is one kind of conditional GAN [20]. As for the dataset, they obtain 42,000 pictures of characters in a Japanese games sales website. And they used a deeplearning based automatic classification method to label character's attributes. In this way, user can control several attributes (e.g. hair color, eye color, eye color, expression) to create own character's face icon. This article is very instructive for us. We learned they data collection method, and expanded its application scope to achieve full-body character generation.

2.2 Metrics

Evaluating the quality of generative models has always been a very difficult task. Because we want the distribution of the generated images to be similar to the dataset and have variable attributes. The exact same picture is not acceptable. Therefore, traditional matrics like PSNR and SSIM cannot be used. Some early researchs used human study to evaluate generative models, which is inefficient and inaccurate.

2.2.1 IS

Inception Score (IS) [30] used entropy to measure the diversity of images. Entropy can be viewed as one kind of randomness. Generally, a value with higly predictable has low entropy. On the contrary, highly unpredictable value's entropy is higher.

y means label and x is generated pictures. In GAN, given an image, we should know its classification easily. That means the conditional probability $p(y|x)$ has low entropy (highly predictable). So, using an classification network (generally, a Inception V3 [31] network pretrained by ImageNET [29]) to identify the generated images and predict $p(y|x)$. It will reflect the qualilty of generated images.

To measure the diversity of images, we can calculate the marginal probability $p(y)$.

Using these two part to compute KL-divergence, we gan get Equation 2.8 .

$$IS(G) = exp(\mathbb{E}_{x \sim p_g} D_{KL}(p(y|x)||p(y))) \tag{2.8}$$

2.2.2 FID

Fréchet Inception Distance (FID) used an classification network (the same as IS, pretrained Inception V3 by ImageNET) to extract featuires from intermediate layer. [6] The FID between ground truth images x and generated images g is calculated as Equation 2.9. Different with IS, lower FID value means better generative quality and diversity.

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}) \tag{2.9}$$

FID is sensitive to mode collapse and more robust to noise than IS. So FID is a better metric for measuring image diversity.

Chapter 3

Face Generation

Standing painting is one kind of illustration. Even photos and illustrations are all pictures, the difference in data distribution between photos and illustrations is quite obvious. Compared to photos, illustrations tend to have fewer colors, but have rich edge information. We can empirically believe that the continuity of the photos is stronger, and the illustrations are more discrete. When performing a two-dimensional image convolution operation, discrete information is often more difficult to learn.

Moreover, the mainstream evaluation metrics FID and IS need to extract features of images by a classfication network. In fact, all of them used pretraine Incetpion v3 by ImageNet (A advanced photos datasets with more than 14 million images). Therefore, it is difficult to determine whether FID and IS can be effective in illustration evaluation.

Since the data distribution of photos and illustrations is so different, we need to design a set of verification experiments to guarantee that GANs can effectivelyh synthesize illustrations and FID is a available matric to evaluate the quality of illustrations.

This preparative experiment will start with simpler anime character' faces generation rather than full-body illustration.

3.1 Proposed Approach

DCGAN, WGAN and WGAN-GP have been introduced in 2.1, they are most classic generative models in the theoretical line of GANs. More of current models are extended from them. We can say that if they can generate illustrations, other advanced structures will also have no problem. By the way, we can also test their performance and the effectiveness of FID.

We will respectively train DCGAN, WGAN and WGAN-GP by two different datasets.

1. CelebFaces Attributes Dataset (CelebA) [28] is an open source dataset with 202,599 number of celebrity face images. It is originally a large-scale face with 40 attribute annotrrations. But in fact, its attribute annotrrations usually not used, as a standard unsupervised dataset for image generation tasks. The dataset has large diversities and quantities, faces of pictures are extracted unified location and the resolution is normalized to 178×218. For ease of use, we further crop them to 128×128.

2. AniFace is our self-built dataset. All of the original illustrations are download from Pixiv. Most illustrations are produced by professional painters. The faces in illustrations are detected by a application called *lbpcascade_ animeface*. This dataset contains 945 faces with normalized resolution of 128×128.

3.2 Model Structures

We used modified DCGAN as our network. The original version of DCGAN is designed to synthesize 64×64 picturess. [22] To output 128×128 pictures, we changed some details of this network. The structures of Generator and Discriminator are shown by Figure 3-1.

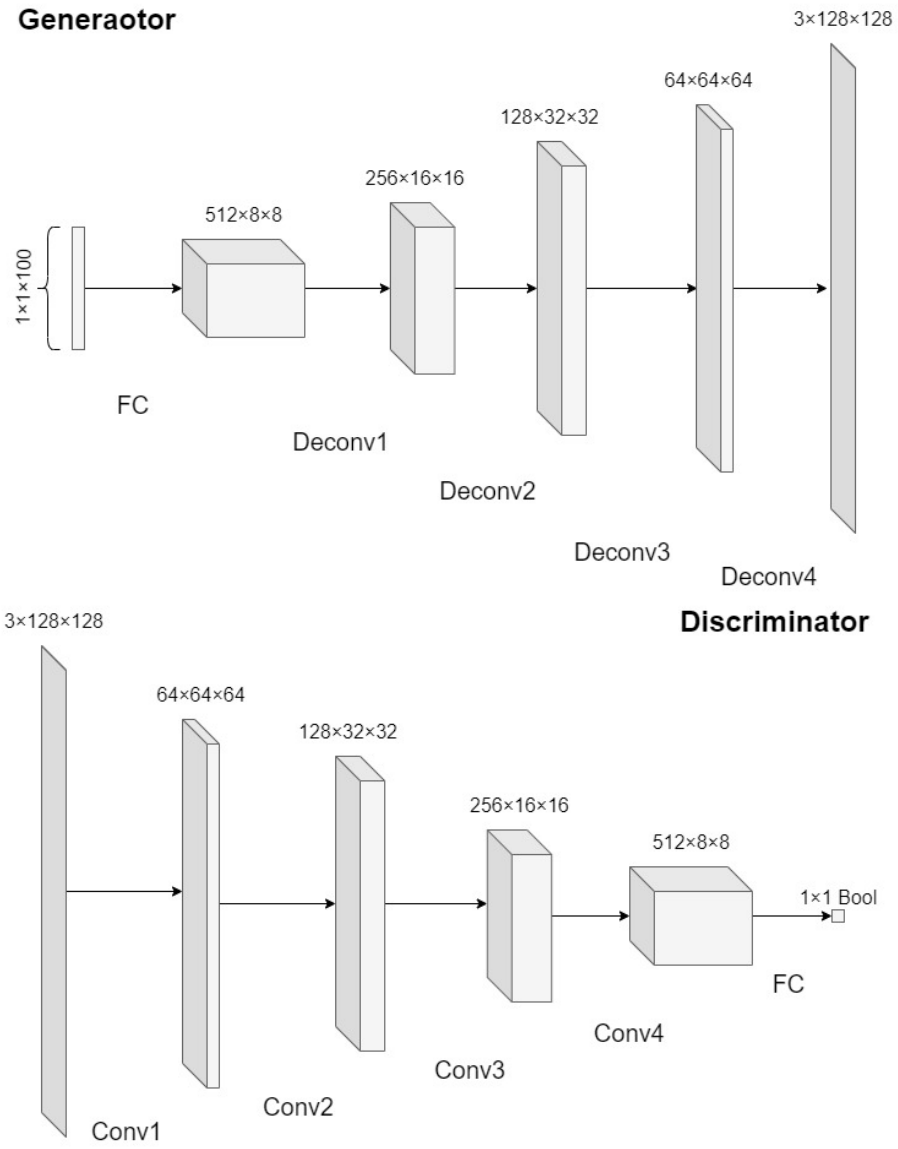

Figure 3-1: Generator and Discriminator of DCGAN

The input latent code z of the Generator is a 100-dimensional distribution noise. After a fully connected layer, it will be transformed into 512 8×8 convolutional kernels. There are 4 upsampling layers of this model.One upsampling layer contains a deconvolution, a activation function and a LeakyReLU.

All deconvolution layers' kernels size is 4×4 , stride is 2 and padding size is 1. With this setting, by Equation 3.2, each deconvolution will double the size of the features, and quickly reach the target size. The activation function of the convolution layer is

LeakyRelu. As for output layer, tanh will be used to limit data range ($0 \sim 255$).

$$size_{conv} = (size_{in} - size_{kernel} + 2 \times padding)/stride + 1 \quad (3.1)$$

$$size_{deconv} = size_{in} \times stride + size_{kernel} - 2 \times padding - stride \quad (3.2)$$

The structure of the Discriminator is almost symmetrical with the Generator. Discriminator output 1 dimensional prediction through 4 downsampling layers and 1 fully connected layer. As for the output layer, DCGAN used Sigmoid as its activation function but WGAN and WGAN-GP did not use activation function. Except for the Discriminator of WGAN-GP used layer normalization [2], all other models used batch normalization [8] as normalized function.

3.3 Training Detail

In all models, we set the learning rate as 0.0002 and batch size as 64. The device we used is a Nvidia RTX 1080. The experimental environment is Tensorflow on win10. Noting that the performance of WGAN and WGAN-GP will get better and better as the training progresses, but mode collapse occures during the training process. So we just take the best result of DCGAN.

In addition, the training process of Discriminator and Generator in WGAN-GP are not synchronized. In fact, every training Discriminator 5 times, Generator just be trained only once. Here, we used training iterations of Generator as standard.

3.4 Result

Figure 3-2 $\sim$ Figure 3-7 show the synthesized faces of GANs. Table 3.1 and Table 3.2 show the numeric results.

We can find that although DCGAN is not as good as WGAN, it is ten times faster than WGAN-GP, the effect is not bad. As for WGAN, the underlying problem caused

it to perform poorly.

Whether in human face generation task and anime face generation task, WGAN-GP shows its amazing synthesize ability. Some faces in Figure 3-4 are difficult to distribution by human eyes. Although synthesized anime faces are not as perfect as human face.

Table 3.1: Results of CelebA

GAN Type	Time	Iterations	FID
DCGAN	1.8h	14400	150.40
WGAN	9.5h	31650	191.80
WGAN-GP	19h	31650	**120.52**

Table 3.2: Results of AniFace

GAN Type	Time	Iterations	FID
DCGAN	0.5h	4000	204.47
WGAN	6h	20000	219.20
WGAN-GP	12h	20000	**145.21**

We can believe that GANs already catch many features of anime face. Be aware that this is trained by a dataset only contains 945 images, 1 / 200 of CelebA.

In conclusion, through experiments, we have proved that GAN is effective in illustrations generation task. It's not much worse than the photoes generation task.

3.5 About FID

An additional experiment will be performed to confirm the efficiency of FID. The result is shown by Talbe 3.3.

Firstly, we randomly picked 945 pictures from CelebA and calculated the FID between AniFace is 217.40. That is mean the FID of 217.40 is a fairly large to distinguish 2 totally different datasets.

Then, we calculated the FID of two different batches in the same dataset. The value reflects the smoothness of dataset.

Table 3.3: Test of Fid

Data(A-B)	Pictures	FID
CelebA-Aniface	945	217.40
2 batch from CelebA	64	2.27
2 batch from AniFace	64	5.32

In conclusion, although FID trained without illustration, it can also evaluate the quality of illustrations.

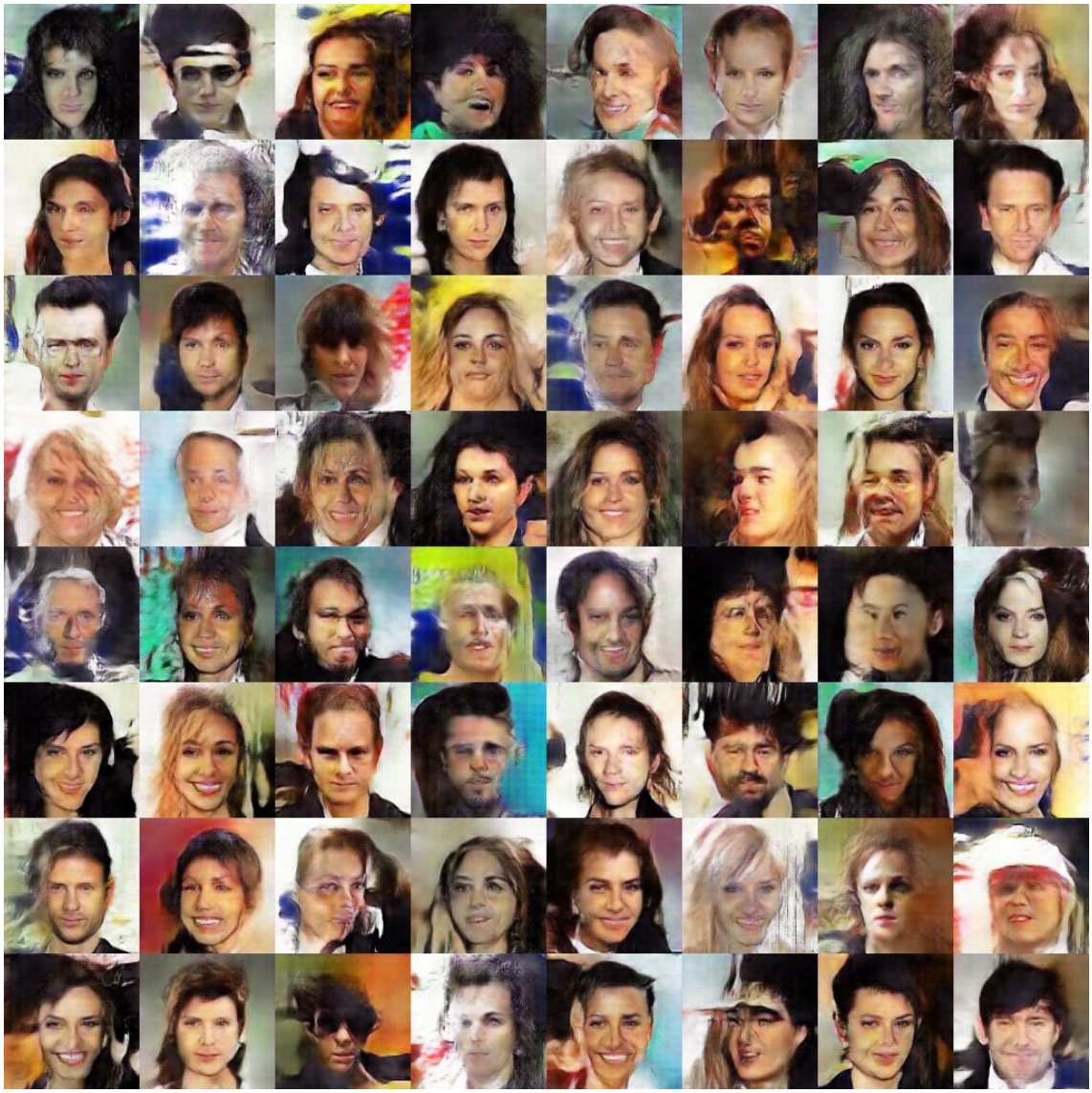

Figure 3-2: Synthesized human faces by DCGAN

Figure 3-3: Synthesized human faces by WGAN

Figure 3-4: Synthesized human faces by WGAN-GP

Figure 3-5: Synthesized anime faces by DCGAN

Figure 3-6: Synthesized anime faces by WGAN

Figure 3-7: Synthesized anime faces by WGAN-GP

Chapter 4

Full-body Generation

In the previous chapter, we have demonstrated the feasibility of GANs in generating illustrations. But we know that full-body illustrations are very different from faces. First, the resolution of the full-body illustration is several times of face. Second, the position of the face is relatively fixed in one picture, which is easy to learn. However, the body illustrations are rich in movements. Third, through the face detector, faces can be extracted from any illustration. It is difficult to obtain a clean standing painting without background.

In order to solve these problems, we need to create a new dataset and look for a powerful generative model.

4.1 Data Collection

Our ideal data set has the following characteristics.

1. As large as possible. The representative open source dataset CelebA has 200,000 images. Another face dataset Flickr-Faces-HQ Dataset (FFHQ) [14] has 60,000 images. Although GANs can work on small-scale datasets with hundreds of images, we want our dataset to be as large as possible to ensure the generation quality.

2. It contains no impurities. In fact, most datasets often do not deal with the back-

ground. We can find flaws in the background of almost all advanced generation models. However, for standing paintings used for game production, unclean backgrounds are unacceptable.

3. The resolution is high enough. The 64×64 face images can also be recognized by human eye. Since the general head-to-body proportions of illustrations is between $5 \sim 8$. We hope to synthesize pictures with resolution of 512×512.

Figure 4-1: Samples of the standing dataset

To collect standing paintings, a directly idea is to extract resources from games.

In fact, we tried this approach initially. We have found several Japanese games, each of them can provide us with hundreds of source illustrations. But a serious problem is that their format is not uniform. Depending on the design style, some games do not require full body standing paintings, only the upper body is used. Another problem is the huge amount of repetitive data. In order to express different expressions and actions, generally there are several differential versions of one standing paintings. Through they be one kind of data augmentation, differential versions increase the weight and affect the balance of the entire dataset. The last problem is that the style in one game is often similar, which also affects the balance of and reduces the diversity of the dataset.

So, inspired by Makegirlsmoe [10], we found *getchu*[1], which is a game retail website, most pictures from the website have resolution of 500×500. We crawled about 60,000 pictures from it and used the following 4 steps to process these raw data.

1. Because the same game may release multiple versions, there is a lot of duplicate data. We selected similar pictures through MD5. The MD5 message-digest algorithm [26] is a hash function widely used to verify data integrity. It can produce a 128-bit hash value for every file. In experience, pictures with Hamming distance of MD5 less than 3 are often very similar. Only one picture in each similar group is kept.

2. Some lower quality pictures were manually removed.

3. Most pictures in this website are watermarked. We removed the 20×100 pixels in the lower right corner of all pictures. This method can process 90% of the pictures.

4. Padding pictures to 512×512.

After these operations, we retain 12,000 high-quality pictures. Figure 4-1 show some samples of the dataset[2].

[1] http://www.getchu.com/

[2] https://drive.google.com/open?id=1kjLcBWpNc59DWhOWJDi3VIHMWXwL2IXX

4.2 Proposed Approach

In recent years, several latest achievements of GANs that can generate high resolution have appeared. Compared to the previous GANs, they can surprisingly synthesize images with a resolution of 1024x1024. The representatives of them are BigGAN [3], MSG-GAN [12], Pro-GAN [13] and StyleGAN [14].

This paper selected StyleGAN as baseline. Compare with other advanced GANs, StyleGAN requires less computing resources. For example, the training of BigGAN requires several weeks on hundreds of graphics cards. However, StyleGAN can be trained on an ordinary computer with only one high-end graphics card. In addition, as its name suggests, StyleGAN has amazing feature: style transform.

The structure of StyleGAN is very different with others. Figure 4-2 shows its generator.

The generator of StyleGAN is a progressive network [13]. A progressive network do not synthesized target image in one step. It firstly generates pictures with low resolutions, and then uses these low-resolution pictures as conditions to generate high-resolution pictures. With this method, training will be more stable. When the memory is insufficient, the bottom layers can have larger batchsize. Each resoluton level of StyleGAN contains two convolutional layer. This generator has 26.2M trainable parameters. For a high resolution network, it is quite streamlined.

Don't like previous GANs, StyleGAN do not have traditional input, the first layer of generator is a vector of constants. Latent code do not input to the generator directly. Before entering the network, the original hidden layer code Z (generally a Gaussian noise) needs to pass through a mapping network. The mapping network is a multi-layer perceptron, encoding Z to an intermediate latent code W. The significance of this operation is to extract meaningful information from noise. W is decomposed into several parts (w1 $\sim$ w16), input each resolution unit.

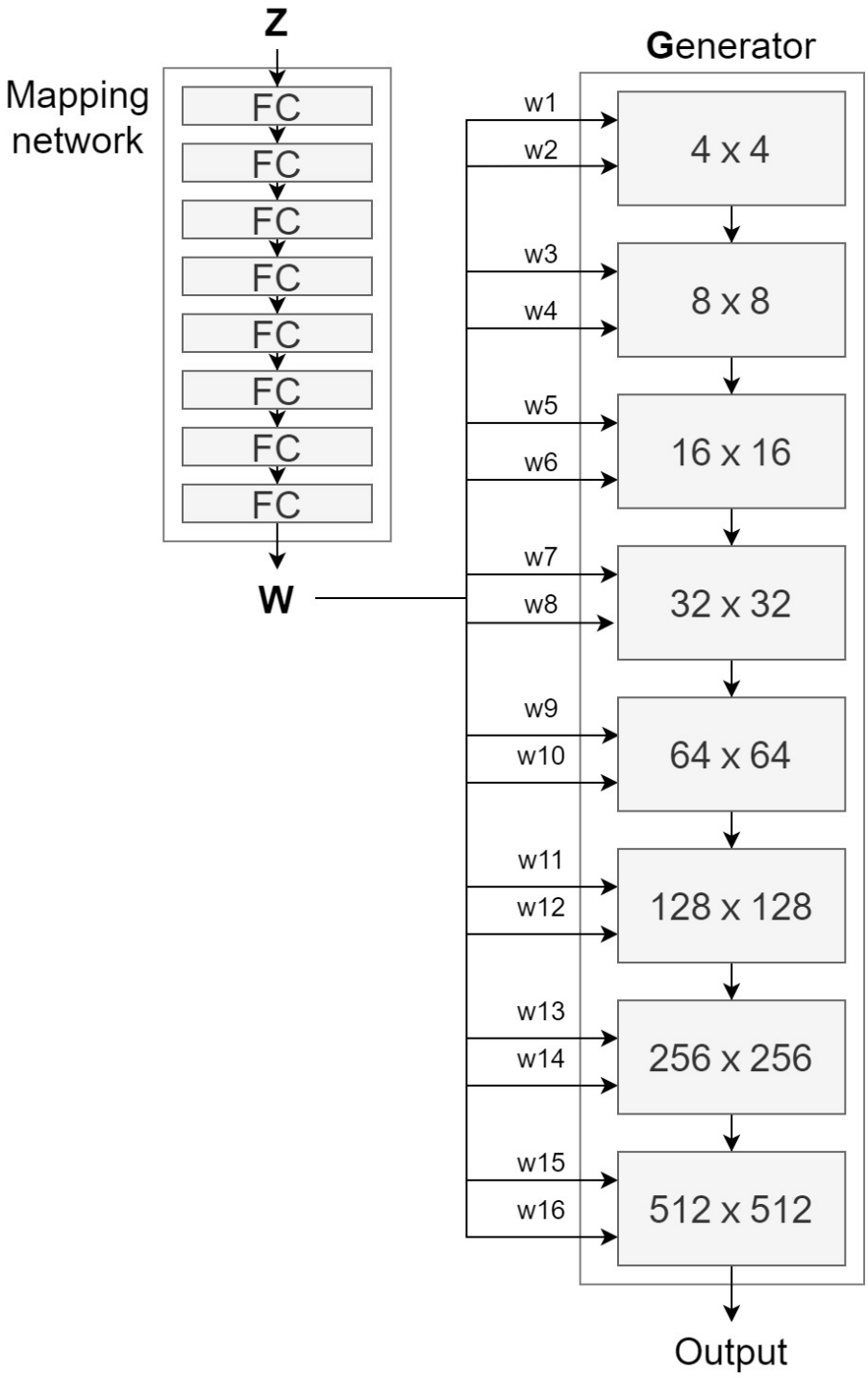

Figure 4-2: The Generator of StyleGAN

4.3 Train Detail

The graphics card we used is one Nvidia RTX 2080-Ti with 11G memory. The experimental environment is Tensorflow on win10. When training, we used dynamic learning rate and batch size in Table 4.1.

Table 4.1: Parameters of Training

Resolution	learning rate	batch size
4×4	0.0015	32
8×8	0.0015	32
16×16	0.0015	16
32×32	0.0015	8
64×64	0.0015	4
128×128	0.0015	4
256×256	0.002	4
512×512	0.003	4

This experiment tested three different variants of StyleGAN. StyleGAN(a) is the baseline StyleGAN as paper [14]. StyleGAN(b) and StyleGAN(c) are improved methods from [15]. StyleGAN(b) do not use growing structure [13], and StyleGAN(c) used a larger network. Since the representative position of WGAN-GP in tranditional GANs, We also trained a 128×128 WGAN-GP as a reference of FID.

All methods are trained with 8000 kimgs, which is approximately equal to 700 epochs. As data augmentation, all pictures are mirrored with a 50% probability before entering the network.

4.4 Result

After long training, we got some interesting result which is shown in Table 4.2. More details can find at our github[3]. The calculation consumption of WGAN-GP is less than 1/16 of StyleGAN(a), but achieved better FID score. Among these StyleGANs, since the growing structure, StyleGAN(a) is fastest. However, the FID of StyleGAN(a) is even worse than WGAN-GP. We know that the FID of 220 can represent two different datasets. As human subjective evaluation, the synthesized samples shown in Figure 4-4 are much better than Figure 4-3. There are two possible reasons. First, FID is invalid on this dataset. Second, the generative diversity of StyleGAN(a) is insufficient, which seriously lowers FID.

Table 4.2: Results of StyleGAN

Model	Resolution	Time	FID
WGAN-GP	128×128	1d 8h	187.31
StyleGAN (a)	512×512	3d 17h	220.77
StyleGAN (b)	512×512	10d 2h	7.28
StyleGAN (c)	512×512	19d 6h	**5.02**

The results of StyleGAN(b) and StyleGAN(c) are more credible. In the same training iterations, StyleGAN(c) spent almost twice as long to win StyleGAN(b) with a slight advantage. But no matter what, as human eyes, results of them seems to be almost the same.

In addition, we can also see that the generated image has some defects. Some of the clothes are weird. And character's hands often synthesized incompletely.

[3]https://github.com/zampie/standing-stylegan

Figure 4-3: Generated samples by WGAN-GP, FID=187.31

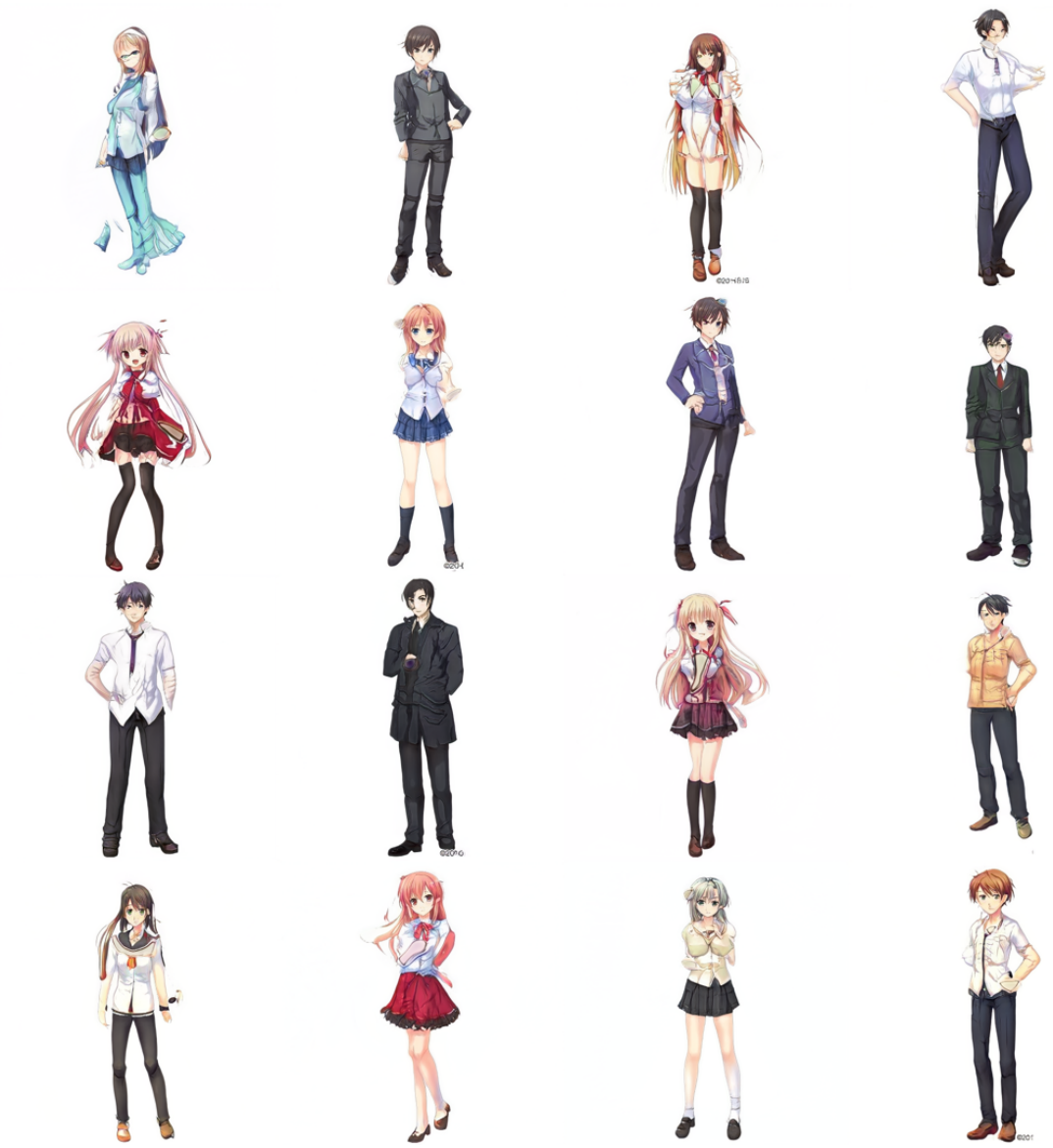

Figure 4-4: Generated samples by StyleGAN(a), FID=220.77

Figure 4-5: Generated samples by StyleGAN(b), FID=7.28

Figure 4-6: Generated samples by StyleGAN(c), FID=5.02

4.5 Style Mixing

Style transformation is an important subject of image processing. It can be realized by non-GAN methods [7, 11, 32]. In GANs, style transformation is often implemented by supervised [9, 27, 33] or semi-supervised [16, 34] models.

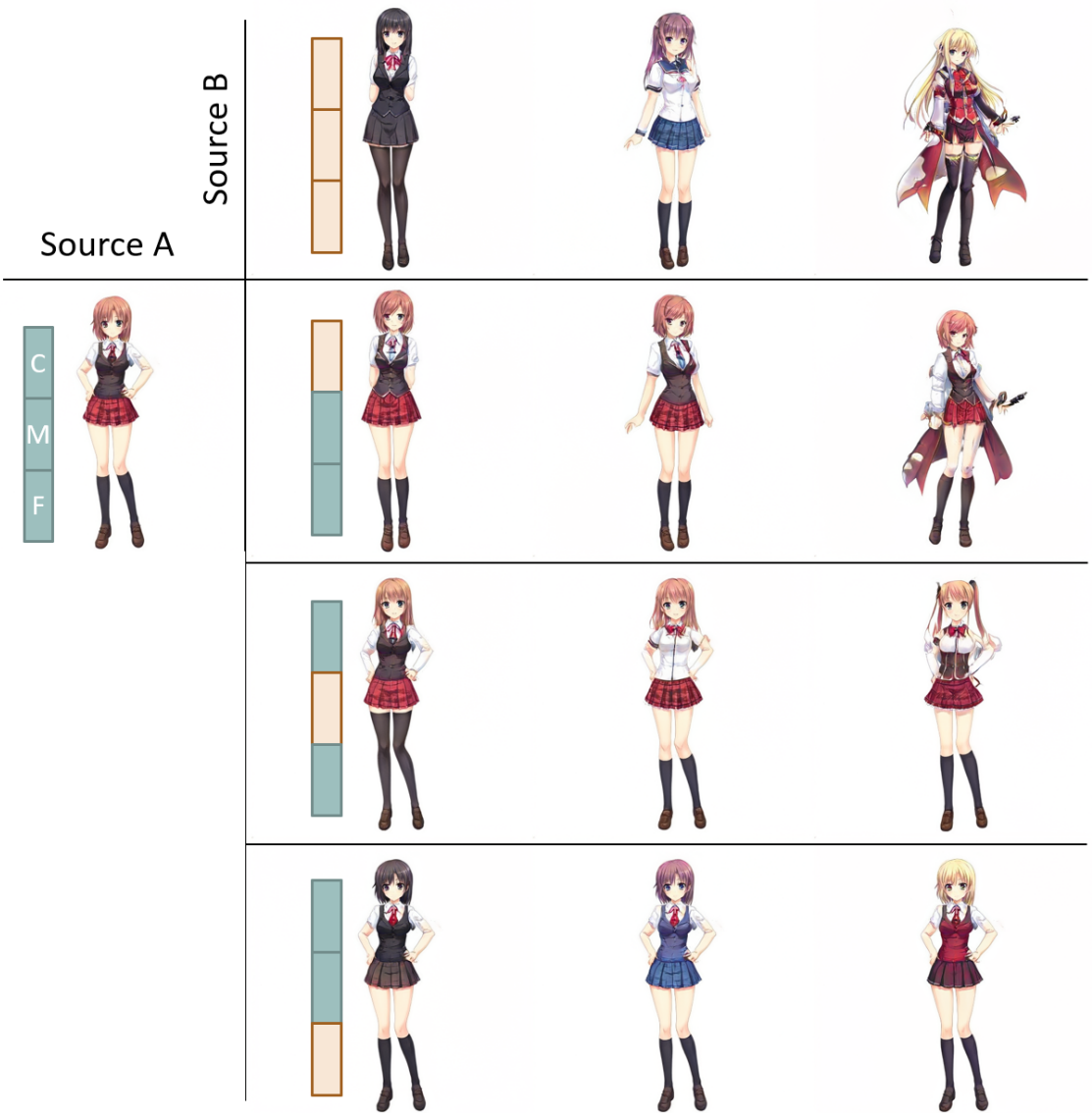

Figure 4-7: Style Mixing

As the name suggests, StyleGAN has the ability of style transformation, even it is an unsupervised model. The ability of StyleGAN depend on its mapping network in Figure 4-2. By manipulating intermediate latend code W, some features of synthesized

pictures can be controlled.

Empirically, W can be divided into 3 parts which represent 3 kinds of style.

1. Coarse style: w1 $\sim$ w6 corresponds $4 \times 4 \sim 16 \times 16$ levels of generator.

2. Middle style: w7 $\sim$ w10 corresponds $32 \times 32 \sim 128 \times 128$ levels of generator.

3. Middle style: w11 $\sim$ w16 corresponds $256 \times 256 \sim 512 \times 512$ levels of generator.

In Figure 4-7, all characters are synthesized by stylegan(c). By replacing a style in source A with a style in source B, 2 characters' attributes will be mixed.

We can find that, the coarse style can control character's actions, the middle style can control character's clothes and the fine style can control overall color of pictures.

We can use this property to control many properties and create customized characters.

4.6 Animation Generation

According to the result of the previous section, we know that chapter's feature can be extracted and changed respectively. So, by controling features of action in standing paintings, we can make some simple animations

In Figure 4-8, we keep the first character's middle style and fine style, interpolate coarse style to other chapters whose poss is different.

Figure 4-8: Synthesized Animation

By this approach, many simple animations be generated in an instant, which has great application potential.

4.7 Encoder of Latend Code

StyleGAN can extract features from synthesized images rather than real illustrations. The key to the problem is to get the intermediate latent code W of real illustrations.

For this, we trained an encoder. The structure of encoder is a 5 layers fully convolutional network. The input of encoder is a $3 \times 512 \times 512$ illustrations, and the expected output is the intermediate latent code W.

Figure 4-9: Ground truth from dataset

Figure 4-10: Reconstructed images by the encoder

Put the pictures in Figure 4-9 to get W, and then put W into stylegan's generator, reconstructed pictures are shown in Figure 4-10.

Maybe the encoder is too small, reconstructed picture is different from the original picture, many features (e.g. gender, clothes, pose) are restored.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This paper discusses the methods and applications of using GANs to generate illustrations.

First, a small face dataset AniFace was created. DCGAN, WGAN and WGAN-GP were trained by the dataset and used to generate animated faces. The preparative experiment verified that GAN can generate illustrations, and FID can also be used to evaluate illustrations.

Then, through crawling and data cleaning, a standing painting dataset was created. By this dataset and StyleGAN, this paper is the first successful research about anime character's full-body standing painting generation.

Finally, by analyzing the middle latent code and separating the character's features, the potential that using GANs to generate animation was discovered.

5.2 Future Work

At present, the quality of the generated standing painting is not perfect. The most significant reason is that the dataset is not big enough, it can also be expanded by the game's resource files. Another reason is that limited to memory, the batch size we used to train StyleGAN's is too small. In subsequent rearch, larger batch size can be used to stabilize the training process.

For wider application of this StyleGAN based standing painting generative model, we should train a more advanced encoder to obtain the intermediate latent code from natural pictures. In this way, the style transformation of all pictures can be achieved.

Chapter 6

Appendix

6.1 More experimental results

Figure 6-1: The averange output of StyleGAN(c)

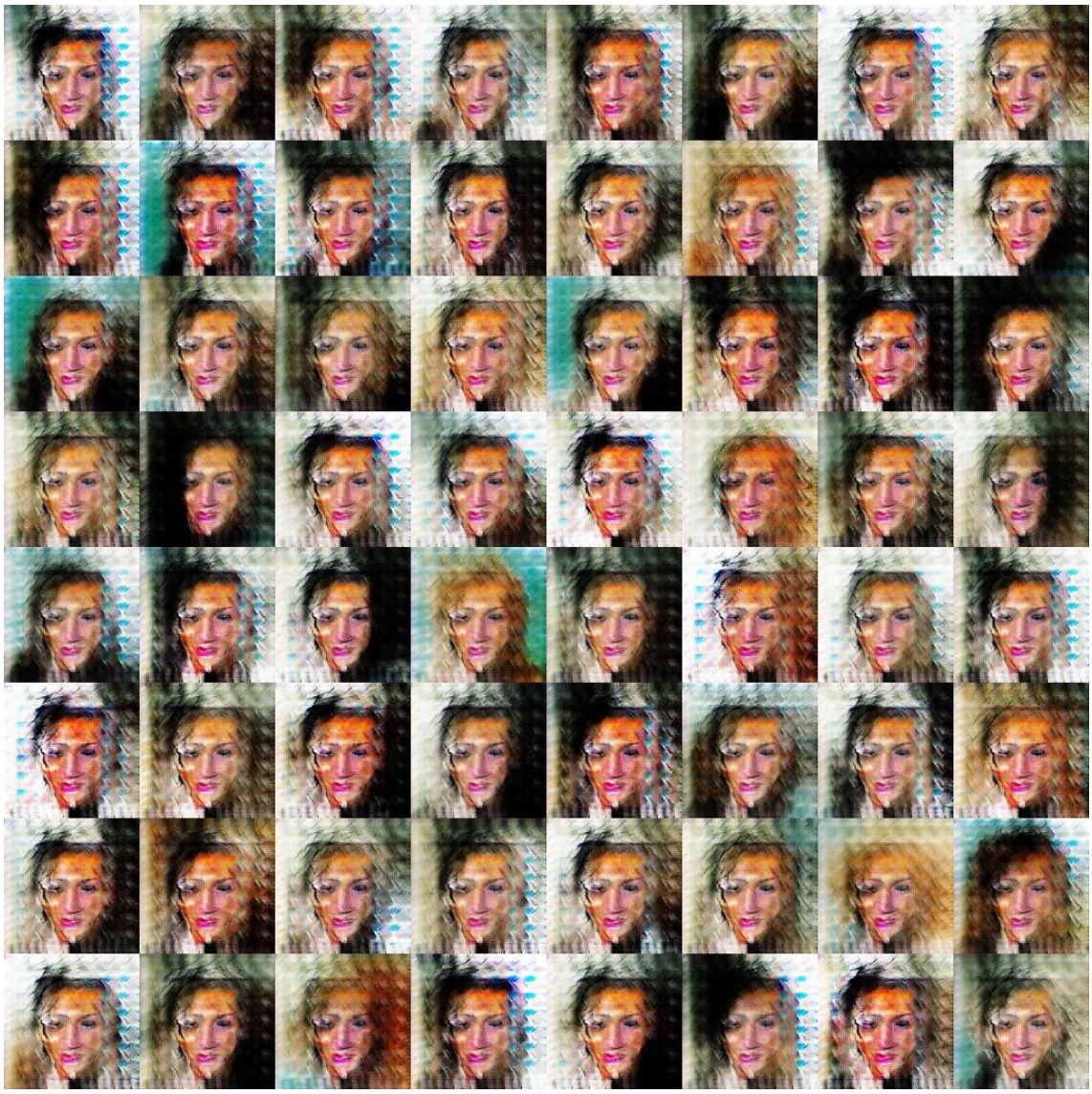

Figure 6-2: Mode collapse of DCGAN in CelebA

Figure 6-3: Mode collapse of StyleGAN in standing painting dataset

6.2 Code about style mixing

```
import os
import pickle
import numpy as np
import PIL.Image
import dnnlib
import dnnlib.tflib as tflib
import time
import pretrained_networks

if __name__ == '__main__':
    root = './'
    synthesis_kwargs =
        dict(output_transform=dict(func=tflib.convert_images_to_uint8,
        nchw_to_nhwc=True), minibatch_size=8)

    tflib.init_tf()
    os.makedirs(root, exist_ok=True)
    ckpt = './ckpt/network-snapshot-007339.pkl'
    _G, _D, Gs = pretrained_networks.load_networks(ckpt)

    seeds = [112, 268, 274]
    imgs = []
    sec = 2
    n = 24 * sec
    truncation_psi = 0.5
    Gs_kwargs = dnnlib.EasyDict()
    Gs_kwargs.output_transform =
        dict(func=tflib.convert_images_to_uint8, nchw_to_nhwc=True)
    Gs_kwargs.randomize_noise = False
    w_avg = Gs.get_var('dlatent_avg') # [component]
```

```
    z_str =
        np.stack([np.random.RandomState(seeds[0]).randn(Gs.input_shape[1])])
w_str = Gs.components.mapping.run(z_str, None) # [seed, layer,
    component]
w_str = w_avg + (w_str - w_avg) * truncation_psi

for seed in seeds[1:]:
    z_end =
        np.stack([np.random.RandomState(seed).randn(Gs.input_shape[1])])
    w_end = Gs.components.mapping.run(z_end, None) # [seed, layer,
        component]
    w_end = w_avg + (w_end - w_avg) * truncation_psi
    w_end[:,6:,:] = w_str[:,6:,:]
    for lamb in np.arange(n) / (n - 1):
        w = (1-lamb)*w_str + lamb*w_end
        img = Gs.components.synthesis.run(w, randomize_noise=False,
            **synthesis_kwargs)
        img = PIL.Image.fromarray(img[0], 'RGB')
        imgs.append(img)
        w_str = w_end
    imgs[0].save(os.path.join(root, 'inter_%s_seeds_%s_model_%s.gif'
        % (time.ctime().replace(' ', '_').replace(':', '_'),
        str(seeds)[1:-1].replace(', ', '_'), ckpt.split('/')[-1])),
        save_all=True, append_images=imgs[1:], optimize=False,
        duration=41, loop=0)
```

Bibliography

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017, arXiv:1701.07875.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016, arXiv:1607.06450.

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018, arXiv:1809.11096.

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

[5] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, 2017, arXiv:1704.00028.

[6] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017, arXiv:1706.08500.

[7] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017, arXiv:1703.06868.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv:1502.03167.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2016, arXiv:1611.07004.

[10] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang. Towards the automatic anime characters creation with generative adversarial networks, 2017, arXiv:1708.05509.

[11] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer: A review, 2017, arXiv:1705.04058.

[12] A. Karnewar and O. Wang. Msg-gan: Multi-scale gradients for generative adversarial networks, 2019, arXiv:1903.06048.

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017, arXiv:1710.10196.

[14] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks, 2018, arXiv:1812.04948.

[15] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan, 2019, arXiv:1912.04958.

[16] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks, 2017, arXiv:1703.05192.

[17] N. Kodali, J. Abernethy, J. Hays, and Z. Kira. On convergence and stability of gans, 2017, arXiv:1705.07215.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2016, arXiv:1609.04802.

[20] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014, arXiv:1411.1784.

[21] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans, 2016, arXiv:1610.09585.

[22] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv:1511.06434.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2015, arXiv:1506.02640.

[24] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger, 2016, arXiv:1612.08242.

[25] J. Redmon and A. Farhadi. Yolov3: An incremental improvement, 2018, arXiv:1804.02767.

[26] R. Rivest and S. Dusse. The md5 message-digest algorithm, 1992.

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015, arXiv:1505.04597.

[28] A. Rozsa, M. Günther, E. M. Rudd, and T. E. Boult. Facial attributes: Accuracy and adversarial robustness. 2018, arXiv:1801.02480. doi:10.1016/j.patrec.2017.10.024.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2014, arXiv:1409.0575.

[30] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016, arXiv:1606.03498.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision, 2015, arXiv:1512.00567.

[32] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. 2017, arXiv:1711.10925. doi:10.1007/s11263-020-01303-4.

[33] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2017, arXiv:1711.11585.

[34] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017, arXiv:1703.10593.