Towards Transfer Learning for GANs

Author: ChenYang(C1896057)

Degree: MSc Advanced Computer Science

Supervisor: Dr Yipeng Qin

Institution: School of Computer Science and Informatics, Cardiff University

Date: December 11, 2020

Abstract

The training of generative adversarial networks requires high costs, such as data sets, time, and computer resources. Some researchers have proposed applying transfer learning techniques to GAN. However, these technologies also have some problems. For this case, this project proposes a combination of existing GAN transfer learning technologies. Its main purpose is to maximize the advantages of these technologies and to make up for their shortcomings to a certain extent. For example, improving the adaptability of the pre-trained GAN to the target domain and reducing the number of learnable parameters to increase the efficiency of transfer learning. After analyzing the transfer learning technology of existing GANs, this project chose three relatively stable methods, including: model fine-tuning, updating scale and shift, and freezing some layers in the network. Then according to the characteristics of these technologies, they will be used in the appropriate position in the network. For technologies not used in this project, this dissertation also has a brief analysis. In addition, when the pre-trained GAN is learning on a small data set, the introduction of data augmentation technology can increase the diversity of the small data set and reduce the possibility of overfitting.

Acknowledgements

I would like to thank my supervisor Dr Yipeng Qin for his professional guidance, without his guidance this project would be impossible.

I would like to thank the module leader of CMT400 Dr Bailin Deng for his help and support.

I would like to thank the lecturer of Applied Machine Learning Dr Jose Camacho Collados, I learned about machine learning and deep learning in his course, without these basic knowledge, I would not be able to start this project.

Commercial limitations

The source domain dataset and target domain dataset in this project are from the same website. According to the author's request, these data sets cannot be used for commercial purposes (Gwylab 2019).

Table of Contents

Chapter 1 Introduction..... 8
1.1 Motivation..... 8
1.2 GANs and Transfer learning of GANs..... 9
1.3 Description of the rest of this dissertation..... 10
1.4 Aim and Objectives..... 11
Chapter 2 Literature Review..... 12
2.1 Domain adaption:..... 12
2.2 Mind2Mind:..... 14
2.3 Batch Statistics Adaptation..... 16
2.4 Freeze the Discriminator..... 20
2.5 AdaFM..... 22
2.6 Conclusions for these papers..... 23
Chapter 3 Problem statement and approach..... 25
3.1 Problem statement..... 25
3.2 Approach..... 26
3.2.1 Model pre-training..... 26
3.2.2 Data set of pre-trained model..... 30
3.2.3 Data augmentation..... 31
3.2.4 Classify the layers of the pre-trained model..... 31
3.2.4 Freeze general part..... 32
3.2.5 Update the scale and shift..... 33
3.2.6 Fine-tune specific part..... 34
3.2.8 Use FID to evaluate results..... 34
Chapter 4 Products..... 35
4.1 Requirement..... 35
4.1.1 Implement model pre-training..... 35
4.1.2 Diversify the data of the target domain..... 36
4.1.3 Use different methods for different parts of the model..... 36
4.2 Design..... 36
4.3 Implementation and results..... 37
4.3.1 Model pre-training..... 37
4.3.2 Data augmentation..... 42
4.3.3 Load pre-trained model..... 43
4.3.4 Model fine-tuning..... 46
4.3.5 Frozen layers ,updating scale and shift..... 48
4.3.6 Compare with other methods..... 54
Chapter 5 Analysis..... 59
5.1 Analysis of model pre-training..... 59
5.2 Analysis of data augmentation..... 59
5.3 Analyze the combination of methods..... 59
5.4 Achieved goals and needs..... 60
5.5 Limitations..... 60

Chapter 6 Conclusions and future work..... 62
6.1 Conclusions..... 62
6.2 Future work.....63
Chapter 7 Reflection..... 64
Reference list:..... 65
Appendices.....66

Table of figures

Figure 1 : Comparison of pre-trained GAN and GAN trained from scratch..... 13
Figure 2 : Decomposition of discriminator and generator.....15
Figure 3 : The training process of Mind2Mind..... 15
Figure 4 : The formula of update the scale and shift..... 17
Figure 5 : Formula of L1 and perceptual losses.....18
Figure 6 : Comparison of the method in the paper with other methods..... 19
Figure 7 : FID scores of frozen discriminator and model fine-tuning..... 20
Figure 8 : Images generated by freeze discriminator and model fine-tuning.. 21
Figure 9 : Comparison of FS and AdaFm.....23
Figure 10 : The calculation formula of AdaFM.....23
Figure 11 : The network structure of DCGAN.....28
Figure 12 : The training process of WGAN.....29
Figure 13 : Loss function of the discriminator in WGAN-GP.....30
Figure 14 : Generator network structure..... 38
Figure 15 : The network structure of the discriminator.....39
Figure 16 : The code of data augmentation.....43
Figure 17 : Load the parameters of the pre-trained generator.....44
Figure 18 : Rewrite the forward function of the generator..... 44
Figure 19 : Get the key of parameters in generator.....45
Figure 20 : Load the parameters of the pre-trained discriminator..... 45
Figure 21 : Rewrite the forward function of the discriminator.....45
Figure 22 : Get the key of parameters in discriminator..... 46
Figure 23 : Freeze four layers in G and D respectively.....46
Figure 24 : Freeze three layers in G and D respectively.....47
Figure 25 : Code to initialize scale and shift..... 48

Table of tables

Table 1 : Comparison of WGAN and WGAN-GP 41
Table 2 : Comparison of the data augmentation and the original image.....43
Table 3 : The results of D(2,1,2), G(2,1,2).....50
Table 4 : The results of D(1,2,2), G(1,2,2).....51
Table 5 : The results of D(1,2,2), G(3,0,2).....52
Table 6 : The results of D(3,0,2), G(1,2,2).....53
Table 7 : FID of these combinations 54
Table 8 : The result of model fine-tuning 55
Table 9 : Freeze three layers of G and D respectively..... 56
Table 10 : Update scale and shift..... 57
Table 11 : Compare FID of combination and other methods.....58

Chapter 1 Introduction

1.1 Motivation

Generative Adversarial Network (GAN) is a relatively difficult model to train in the field of deep learning. At present, it mainly includes the following difficulties:

1. Generator gradient vanishing

When the discriminator is very accurate, the loss of the discriminator converges to 0 quickly, so it can not provide a reliable path for the gradient of the generator to continue to update that resulting in the disappearance of the generator gradient. At the beginning of training, the fake data generated by GAN is far from the real data, so there is almost no overlap between the two distributions. At this time, the discriminator can quickly learn and distinguish between real data and fake data, that resulting in the gradient of the generator unable to continue to update or even disappear.

2. Sampling calculation distance problem

In GAN, the distance between distributions estimated by sampling is not equal to the true distance between two distributions. If the distance information guiding the generator to learn is biased, it is likely that the implicit probability distribution defined by the generator can not be approximated to the essential probability distribution of the data set.

3. Mode collapse

In practical applications, complete pattern collapse is rare, and local pattern collapse is very common. Local mode collapse means that the generator makes different pictures contain the same color or texture theme, Or different pictures contain different parts of the same object.

For this case, the main motivation of this project is to reduce the difficulty of GAN training, such as: reducing the data required for GAN training, improving the performance of GAN. Transfer learning can reuse existing models, so that

these pre-trained models not only have good results in the original task, but also have good performance in other tasks. Therefore, transfer learning also makes those pre-trained models more versatile which also makes the work of the developers of these models more meaningful.

1.2 GANs and Transfer learning of GANs

The Generative Adversarial Network (GAN) is a groundbreaking generative model proposed by Goodfellow et al in 2014. It consists of two networks, a generator and a discriminator, and introduces the idea of confrontation between the generator and the discriminator during training (Goodfellow et al. 2014). But this model has some problems, such as model collapse, unstable training, etc. Since GAN was invented, the improvement of GAN has become one of the hot research topics in the field of deep learning, and more and more GAN variants have appeared, such as DCGAN, WGAN, BigGAN, StyleGAN, etc. Although these improvements have solved the problems of unstable training and mode collapse to a certain extent, these models also have some other problems.

The training of these models requires too much data and computer resources, and even some models require several weeks to complete training with multiple GPUs. In order to solve these issues, some studies have proposed to apply transfer learning to GANs, which can greatly reduce the cost of GAN training. Transfer learning, as the name implies, is to transfer the parameters of the trained model (pre training model) to the new model. It can improve the training efficiency of new models. Considering that most of the data or tasks are related, we can share the model parameters (which can be understood as the knowledge learned by the model) to the new model in a certain way, so as to speed up and optimize the learning efficiency of the model, without learning from scratch. Among them, there are some ways to implement transfer learning (Wei 2017):

1. Fine tuning: This is the most commonly used transfer learning method which can retain some prior knowledge. It needs to load the weights of the pre-trained model, and then train in the data set of the target domain.

2. Freeze layers: Freeze part of the convolution layer (usually the most convolution layer close to the input, because these layers retain a large amount of underlying information) of the pre training model, and train the remaining convolution layer (usually the partial convolution layer close to the output) and the fully connected layer.

However, transfer learning is widely used in image recognition and image classification, and there are relatively few researches on transfer learning in GANs. Therefore, it is necessary to full use of existing technology. For this situation,the main purpose of this paper is to analyze the existing transfer learning methods of GANs and then propose a combination of these approach which can improve the adaptability of the model to the target domain and reduce the amount of parameter calculation. In general, the main task of this project is to find a balance between model performance and training efficiency. In addition, in order to further reduce training costs, this project will complete the migration learning of GAN on a small data set. At the same time, the small data set may cause the model to overfit, which is also one of the issues that this project needs to consider.

1.3 Description of the rest of this dissertation

1. Aim and Objectives: determine the goal of this project, and alternative directions

2. Background material: the background knowledge needed for this project and analyze related papers on the transfer learning of GANs

3. Problem statement and approach: Mainly describe the problems to be solved in this project and the difficulty of these problems. Then this section also proposes some solutions to these problems.

4. Product: This chapter includes requirements, code implementation and result analysis.

5. Analysis: Analysis of the results and limitations of this project.

6. Conclusion and future work: Some conclusions based on the experiment of this project and plans for future work.

7. Reflections: Reflections summarized in this project.

1.4 Aim and Objectives

The main purpose of this project is to implement the transfer learning of GANs after in-depth research on the principles of transfer learning and GAN. Although there are some existing studies in this area, some improvements are still needed.

Therefore, the following objectives can be determined:

1. Understand the principle of GAN transfer learning through the existing literature, and analyze the methods given in these papers.Then implement GAN transfer learning on a small data set

2. Obviously, transfer learning requires a pre-trained model. When i choose a pre-training model, several issues need to be considered: whether the pre-training model is open source, the source domain data set used by the pre-training model, and the deep learning framework used by the pre-training model.

3. For small data sets, use appropriate data augmentation techniques to reduce the risk of overfitting.

4.Improve the method of GAN transfer learning and make some innovations. The possible improvement directions is: combine existing transfer learning methods to maximize their advantages and then prove that a combination of methods is better than using these methods alone.

For this part, the objectives 4 is exploratory which will update according to the experimental results in the future work. In addition, objectives 2 will have two

choices: find a suitable pre-trained model in the open source library or platform and complete the model pre-training independently in this project, depending on the actual situation and project requirements.

Chapter 2 Literature Review

Currently, transfer learning for GANs is still an open field, especially when the target domain is a small sample data set. Its main idea is to use a pre-trained GAN to implement transfer learning on target domain with little data (100 or 1000 pictures that far below the amount of data required to retrain a new GAN). Therefore, transfer learning for GANs mainly solves the problem of retraining a new GAN that consumes much time and computer resources. But too small a data set can also cause some problems: such as model collapse, overfitting, lack of diversity in generated results etc. Some papers have proposed solutions and in this section, I will analyzed the following methods and made innovations based on some conclusions.

2.1 Domain adaption:

For the GAN (generative adversarial network), If you train from scratch, you need a large data set and much time to generate high-quality images. Yang Xing Wang et al. used the idea of transfer learning and applied the pre training model to the GAN, which effectively solved this problem. They came to the following conclusion (Yaxing et al. 2018):

1. They evaluated several GANs transfer learning configurations, and the results showed that in the case of limited data, using a pre-trained network can effectively accelerate the GANs training process and provide some useful prior knowledge.
2. They proved that the relationship between the source domain and the target domain can affect the results of transfer learning.

3. They evaluated the transfer of two commonly used methods from unconditional GANs to conditional GANs, and proved that conditional GAN can also implement transfer learning.

The main idea of the paper is to apply domain adaptation to the generator, which can improve the effect of the generator and significantly reduce the training time (Yaxing et al. 2018). It is necessary to choose a suitable GAN under this experimental background, because so far there have been a large number of improved versions of GAN, such as: DCGAN, WGAN, WGAN-GP, etc. Finally, in order to study the domain transfer effect of GAN, the authors used the network structure of WGAN-GP. In the paper, the authors used 1000 images from the LSUN bedroom dataset as the target dataset and Image Net as the source dataset (according to previous experiments, the GAN trained on the LSUN bedroom used 3 million images). Furthermore, the author also tried other different combinations of source and target data sets. The following figure is the experimental result of the paper, and compared with the result of training GAN from scratch (Yaxing et al. 2018).

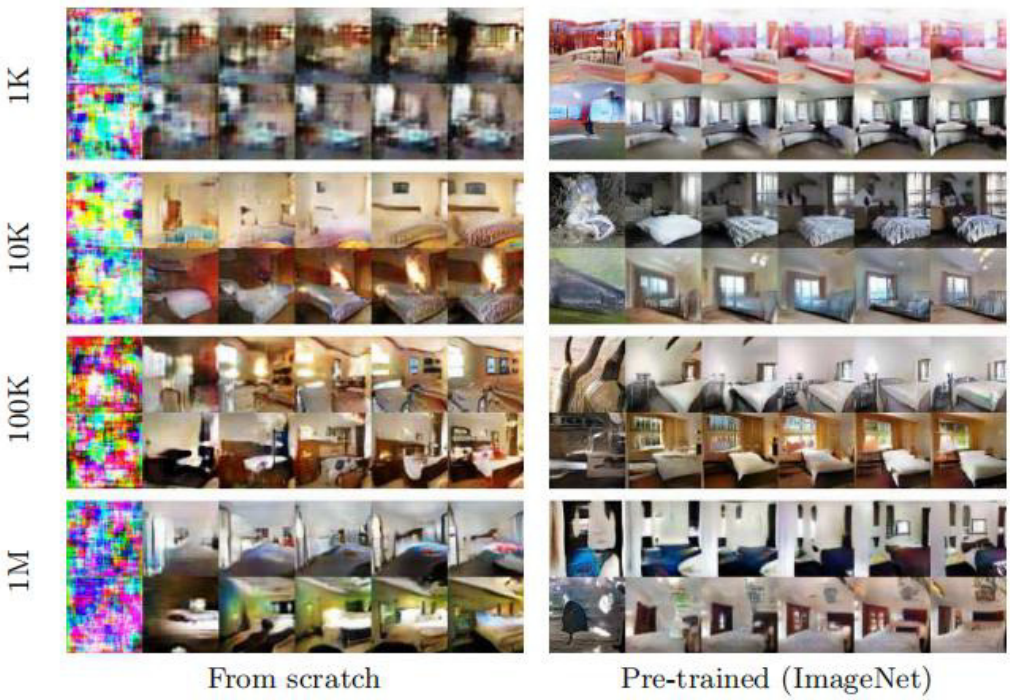


Figure 1: Comparison of pre-trained GAN and GAN trained from scratch

It can be seen from Figure 1 that the pre training model greatly speeds up the convergence speed of GAN . Next, the experiments shows that both the source domain and the target domain will have an impact for the transfer learning of GAN, factors causing these effects include the size, density, diversity of the source domain, and the relative distance between the source domain and the target domain (Yaxing et al. 2018). These factors will affect whether the new network after transfer learning can inherit knowledge which has been learned in the pre-trained network. In addition, the experiment has some additional findings, such as the generator network will start to transfer after the discriminator completes the transfer learning (Yaxing et al. 2018). However, this method has some limitations. For example, if the similarity between the source domain and the target domain is not high or the diversity and density of the source domain samples are insufficient, the effect of transfer learning may be affected.

2.2 Mind2Mind:

Ya¨el Fr´egier and Jean-Baptiste Gouray invented a new method which called Mind2Mind and successfully implemented the transfer learning of Wasserstein GAN in 2019. At first i will briefly explain the main idea of Mind2Mind which divide the discriminator and generator into two parts respectively, for example, a normal neural network can be divided into c0 and c1, similarly, the source domain pre-trained generator g is composed of g0 (mathematically that $g = g0 \circ g1$) and g1, and the generator d is composed of d0 and d1 (mathematically that $d = d0 \circ d1$) (Ya¨el and Jean-Baptiste 2019). The decomposition of the generator and discriminator network is shown in the figure below (Ya¨el and Jean-Baptiste 2019).

$$
\begin{tikzcd}
 & M \arrow[dr, "g_1"] & & M \arrow[dr, "c_0"] & \\
Z \arrow[ur, "g_0"] \arrow[rr, "g"] & & \chi \arrow[ur, "c_1"] \arrow[rr, "c"] & & \mathbb{K}.
\end{tikzcd}
$$

Figure 2: Decomposition of discriminator and generator

Experiments show that the low-level d1 deals with low-level features of the data (usually shared by similar data sets), but the high-level d0 is responsible for high-level features that are very specific to each data set and low-level features often take much time in the session during training (Ya¨el and Jean-Baptiste 2019). Similarly, some of the layers of the generator also represent the unique features of the data set. However, different from discriminator, these layers are located at the beginning of the network (g0) (Ya¨el and Jean-Baptiste 2019). Therefore, we only need to retrain g0 and d0 in transfer learning to make the GAN converge quickly in the target domain. In the paper, the authors give a detailed training method and algorithm for mind2mind (Ya¨el and Jean-Baptiste 2019).

Algorithm 1 Auto-encoding.

Require: : α, the learning rate, b, the batch size, $\mathcal{D}$, a dataset, φ and θ the initial parameters of the encoder c_1 and of the decoder g_1.
while φ, θ has not converged **do**
  Sample $\{x^{(i)}\}_{i=1}^{b} \sim \mathbb{P}_{\mathcal{D}}$ a batch from the dataset $\mathcal{D}$.
  Update c_1 and g_1 by descending L_{AE}.
end while

Algorithm 2 MindGAN transfer learning.

Require: : α, the learning rate, b, the batch size, n, the number of iterations of the critic per generator iteration, $\mathcal{D}'$, a dataset, φ' and θ' the initial parameters of the critic c_0' and of the generator g_0'.
Compute $c_1(\mathcal{D}')$.
while θ' has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{m^{(i)}\}_{i=1}^{b} \sim c_{1\sharp}\mathbb{P}_{\mathcal{D}'}$ a batch from $c_1(\mathcal{D}')$.
    Sample $\{z^{(i)}\}_{i=1}^{b} \sim \mathbb{P}_Z$ a batch of prior samples.
    Update c_0' by descending L_c.
  end for
  Sample $\{z^{(i)}\}_{i=1}^{b} \sim \mathbb{P}_Z$ a batch of prior samples.
  Update g_0' by descending $-L_g$.
end while
return $g_1 \circ g_0'$.

Figure 3: The training process of Mind2Mind

It is divided into two steps: The first step is to train the auto-encoder (c1, g1) in

the source domain, which is equivalent to the process of transfer learning and training, allowing the auto-encoder to learn prior knowledge in a huge source data set. The second step is to pass the data set of the target domain through c1 (the encoder in the auto-encoder), then train a Mind GAN (g_t0, c_t0) on the coded data c1 (target domain), and combine this newly trained Mind GAN with the previously pre-trained auto-encoder. So we can get a new GAN whose generator consists of c_t0 and c1 (mathematically that $c_t = c_t0 \circ c1$) and whose discriminator is composed of g_t0 and g1 (mathematically that $g_t = g_t0 \circ g1$) (Ya¨el and Jean-Baptiste 2019). Finally, I will introduce some configurations of the experiment as a reference, because as mentioned above, different GAN network architectures and different data set combinations will affect the results of the experiment. Similarly, the experiment in the paper still chooses the network structure of WGAN(g,c), which means that in the field of GAN transfer learning, researchers tend to use WGAN and WGAN-GP. In addition, authors divides WGAN into two parts: the encoder C1 of the autoencoder (g1, g1) and the decoder G1 are the convolution part of WGAN (g, c) and the generator g_t0 and discriminator c_t0 of MindGAN (g_t0, c_t0) are the fully connected layers of WGAN (g, c) (Ya¨el and Jean-Baptiste 2019). The experiment of the paper uses four different data sets, and uses different combinations of source and target domains, including MNIST, KMNIST, FashionMNIST and NotMNIST, because these data sets can start GAN transfer learning without data pre-processing (Ya¨el and Jean-Baptiste 2019).

2.3 Batch Statistics Adaptation

Atsuhiro Noguchi and Tatsuya Harada proposed another method to implement GAN transfer learning in 2019. Its main purpose is to further reduce the data set of the target domain (100 pictures). Because the traditional generative model obviously cannot train a high-quality generator with a data set of 100 images and it is very difficult to collect and organize a large number of data

sets in some special fields.In addition, if you want to produce a high-quality data set, you not only need to collect a large number of samples, but also consume much time to complete data cleaning. For this situation, Atsuhiro Noguchi and Tatsuya Harada improved the transfer learning of GAN and implement the pre-trained model to generate images from small data sets. At first, the authors proved that even when the model is fine-tuned with a data set in a domain completely different from the data set of the pre-trained model, the performance of the model is often better than training from scratch (Atsuhiro and Tatsuya 2019). Because the generative network transfers the prior knowledge learned from the source domain to the target domain that means the main factor to improve performance is these prior knowledge. Based on this theory, the authors take how to retain these prior knowledge more precisely as their main task. Then the experimental results show that the scale and shift parameters of batch statistics in the generator are important factors that affect whether prior knowledge can be successfully transferred to the target domain. Then I will briefly describe the filter selection method used in the paper. The convolution operation can be regarded as a combination of filters that convert a three-dimensional tensor to a scalar, so the number of filters is equal to the number of output channels of the convolutional layer. Therefore, applying scale and shift to the convolution operation is shown in the following formula (Atsuhiro and Tatsuya 2019).

$$
\begin{aligned}
&conv(x; W) \cdot \gamma + \beta \\
&= conv(x; W \cdot \gamma + \beta) \\
&= conv(x; \{\gamma_1 W_1 + \beta_1, ..., \gamma_{c_{out}} W_{c_{out}} + \beta_{c_{out}}\})
\end{aligned}
$$

Figure 4: The formula of update the scale and shift

W in the formula is a four-dimensional tensor, and its value is the weight of the convolution. The scale(γ) represents the activation strength of the filter of each

convolution and the shift(β) is the bias which is equivalent to the activation threshold of the convolution kernel . Different from the previous two papers, this experiment chose SNGAN pre-trained on Image Net that means that the pre-trained model has learned prior knowledge of multiple domains.

Moreover, based on the updated scale and shift, the authors propose another method which is to use only the generator of the pre-trained model and then use VGG16 to discriminate the generated pictures. Due to changes in the GAN training strategy, the authors change the loss function (L1 and perceptual losses). The following is the loss function given by the author in the paper (Atsuhiro and Tatsuya 2019).

$$
\begin{aligned}
L = & \sum_i \frac{1}{c_x h_x w_x} ||x_i - G_{Adapt}(z_i + \epsilon)||_1 \\
& + \sum_i \sum_{l \in layers} \frac{\lambda_C^l}{c_l h_l w_l} ||C^{(l)}(x_i) - C^{(l)}(G_{Adapt}(z_i + \epsilon))|| \\
& + \lambda_z (\sum_j^k \frac{1}{d_z} \min_i ||z_i - r_j||_2^2 + \sum_i^b \frac{1}{d_z} \min_j ||z_i - r_j||_2^2) \\
& + \lambda_{\gamma,\beta} \sum_l \frac{1}{d_{\gamma,\beta}^l} (||\gamma_l - 1||_2^2 + ||\beta_l||_2^2) \qquad (3)
\end{aligned}
$$

Figure 5: Formula of L1 and perceptual losses

In the experiment, authors selected three target domain datasets, including FFHQ dataset, Oxford 102 flower dataset and an anime face dataset (from www.gwern.net/Danbooru2018) (Atsuhiro and Tatsuya 2019). In order to prove the success of their method, the authors used their method to compare several other GAN transfer learning methods. The experimental results are shown in the figure below (Atsuhiro and Tatsuya 2019).

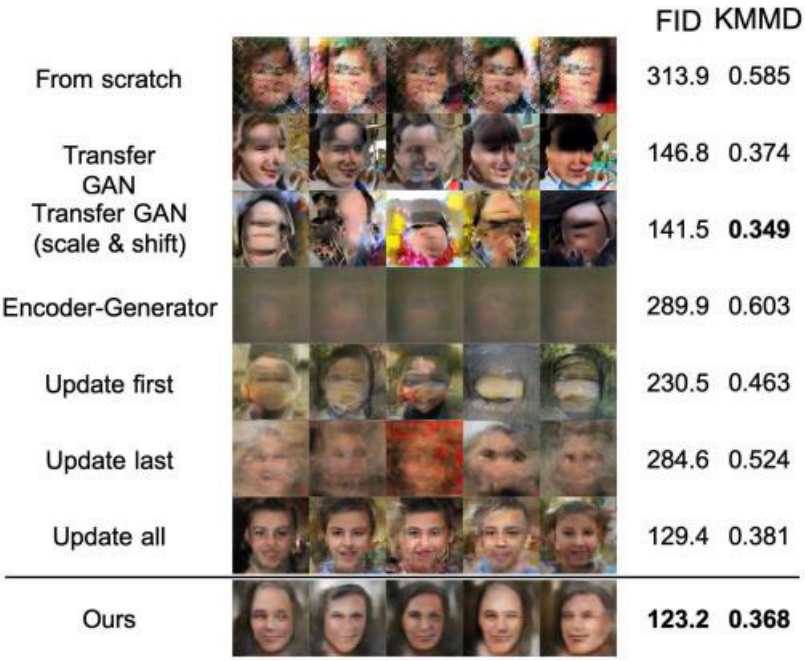


Figure 6: Comparison of the method in the paper with other methods

The results show that this method generates relatively high-quality pictures on the target domain of 100 pictures. Finally, the authors of the paper discussed the choice of source domain data set. They used Image Net, LSUN bedroom data set and face data set respectively (Atsuhiro and Tatsuya 2019). And the experimental results show that the generator pre-trained on the Image Net dataset has better performance in transfer learning. This result also proves that choosing a suitable source domain data set can improve the efficiency of GANs transfer learning. Overview, fixing the convolution kernel in transfer learning and updating the scale and shift can make the pre-trained convolution kernel better adapt to the target domain. However, I think that the method of using only pre-trained model generators in transfer learning proposed by the authors of this paper is not suitable for all situations. This method has the following problems:

1. the problem of coordination between VGG16 and pre-trained generators and if the target domain is not in the training set of VGG16, it possibly can not accurately extract the features of the target domain.

2. If the size(pixel) of the generated picture is different from the size(pixel) of the input VGG16, the generated pictures will need to be up-sampled or down-sampled. These operations may lose the features of the pictures.
3. This method does not make the generator and VGG-16 confront each other.

2.4 Freeze the Discriminator

Sangwoo Mo KAIST et al have proposed a basic and simple method of GANs transfer learning in 2020. Similarly, the main purpose of the authors is to reduce the problem of GANs requiring a large number of data and consuming much computer resources during training (Ya¨el and Jean-Baptiste 2019). In this paper, a simple and effective GANS transfer learning method is proposed which is freeze the discriminator. The experimental results are shown in the figure below (SangwooMo et al. 2020).

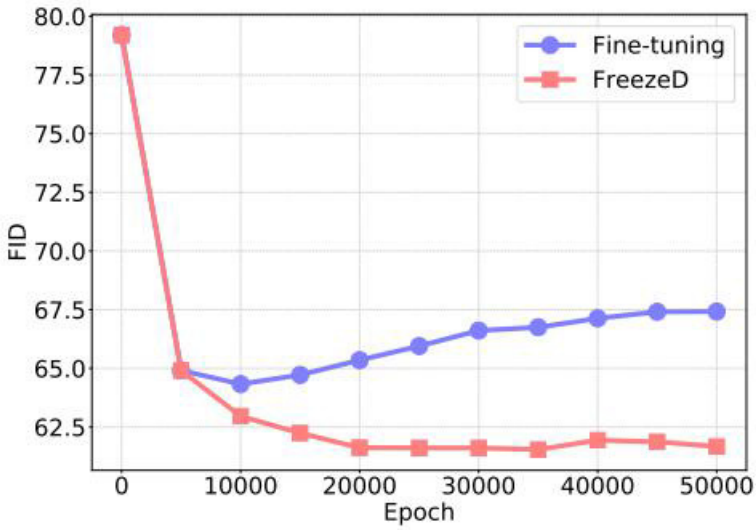


Figure 7: FID scores of frozen discriminator and model fine-tuning

(a) Flower (fine-tuning) (b) Flower (FreezeD)

Figure 8: Images generated by freeze discriminator and model fine-tuning

According to the above results, i can find that the performance of fine-tuning the model while freezing the discriminator in the GAN is significantly improved than the performance of direct fine-tuning. Moreover, freezing the lower network (feature extractor in the discriminator) and then fine-tuning the upper network (classifier in discriminator) has been widely used in image classification. First of all, the author of paper proves that this method is also applicable to transfer learning of GANs and set an appropriate baseline for the transfer learning of GANs(SangwooMo et al. 2020). Then, the authors also describe the advantages and disadvantages of several other transfer learning methods .

Model fine-tuning: The most commonly used and simplest method, but there is a risk of overfitting (SangwooMo et al. 2020).

Scale/shift: This method tends to show poor results, especially when there is a significant difference between the distribution of the source domain and the target domain (SangwooMo et al. 2020). However, this method can be used to enhance the adaptability of the pre-trained model to the target domain when the source domain and target domain are similar.

Generative latent optimization (GLO): Although this method improves stability,

it tends to produce blurry images due to lack of confrontational loss (SangwooMo et al. 2020).

Next, the authors not only use experiments to prove that the method of freezing the discriminator is superior to the above-mentioned method, but also made two improvements to the baseline (freeze D) as future research directions. The first one is L2-SP, bases on the fine-tuning of the model, L2-SP restricts the target model to prevent the target model from moving away from the source model. But this improvement did not achieve satisfactory results. The second improvement is feature distillation, which is also one of the popular methods in transfer learning and achieves similar results to the freeze discriminator (SangwooMo et al. 2020).

2.5 AdaFM

Miaoyun Zhao et al. proposed a new technology in 2020, which allows pre-trained GANs to adapt to more domains (Miaoyun et al. 2020).This technique is similar to the Filter Selection(FS), both of them are used in the general part of GAN. The purpose of this technique is that allow Adaptive Filter Modulation (AdaFM) modify the statistics of the convolution filter to transfer the general part (Miaoyun et al. 2020). On the other hand, the difference is that the trainable parameters introduced in AdaFM are used in the input dimensions and output dimensions of the convolutional layer that means the scale and shift in AdaFM are different from FS. Therefore, this approach may improve the adaptability of the pre-trained model to the target domain. In order to show the difference between FS and AdaFM more intuitively, the authors of this paper provide the following figure (Miaoyun et al. 2020).

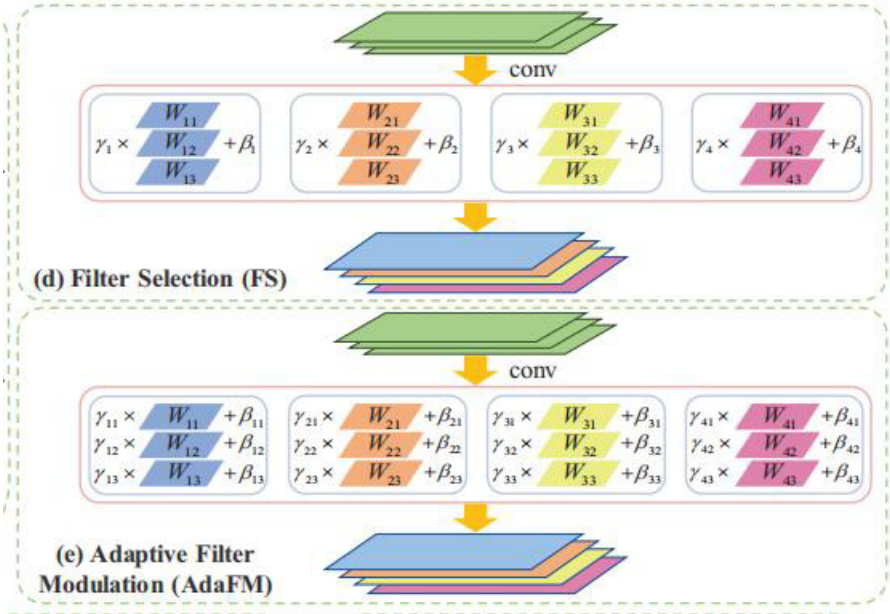


Figure 9: Comparison of FS and AdaFm

Similarly, the authors provided the calculation formula of AdaFM in the paper (Miaoyun et al. 2020).

$$\mathbf{W}_{i,j,:,:}^{\text{AdaFM}} = \gamma_{i,j}\mathbf{W}_{i,j,:,:} + \beta_{i,j},$$

Figure 10: The calculation formula of AdaFM

However, if AdaFM is used in the general part of the model, it will also increase the amount of calculation. That means more time cost and consumption of computer resources. Especially in some complex GANs, this problem will be more serious, such as BigGAN, StyleGAN etc. From the perspective of learnable parameters, this method is between fine-tuning the model and freezing the layers. Therefore, this technique needs to be used at the appropriate layers during transfer learning.

2.6 Conclusions for these papers

According to the above papers, I can get the following conclusions. For the

transfer learning of GANs, directly import the data set of the target domain, and then fine-tune the model is the most basic and most commonly used method. This method can effectively transfer the prior knowledge in the pre-training model to the target domain. Furthermore, the authors of the above papers have improved this basic method from different directions. Therefore, for the experiment of this paper, I will pay attention to some factors, such as choosing a suitable pre-trained GAN, the relationship between the target domain and the source domain, the diversity of the source domain data set, etc. According to the conclusions given in the above paper, in the transfer learning of GAN, the discriminator is more important than the generator, so improving the efficiency of the discriminator can improve the efficiency of transfer learning (Yaxing et al. 2018). Moreover, both the generator and the discriminator retain some common features, which are applicable to both the source domain and the target domain. In most cases, they are located in the upper layer of the discriminator and the lower layer of the generator (Ya¨el and Jean-Baptiste 2019). But there are still some problems to be solved when the target domain has only a small amount of data, include how to train the model more stably, and how to reduce the number of iterations to achieve rapid convergence. In addition, the main task in this project is to implement GAN transfer learning on a small data set. The advantage of a small data set is that it saves the cost of collecting data and the training speed of the model on a small data set is faster than training on a large data set, but it often leads to overfitting. Therefore, I decided to do some special treatment on the small data set of the target domain, like Data augmentation. Finally, i will also experiment with some new ideas in this project, such as combine the above methods in order to maximize the advantages of these methods.

Chapter 3 Problem statement and approach

The main content of this chapter includes the problem statement, project components, and optional experimental methods. The problem statement is mainly based on the research goals. The project components include data augmentation modules, pre-training models, and transfer learning modules. Finally, the feasibility of each alternative method is verified through experiments.

3.1 Problem statement

As described in the previous sections, although the Generative Adversarial Network (GAN) can generate enough fake pictures, it requires too much time and computer resources. These factors make personal research and enthusiasts unable to train a GAN from scratch according to their requirements. This situation also hinders the development of GAN. Therefore, in order to solve this problem, it is necessary to improve the training efficiency and accelerate the convergence of the model by transfer learning. However, most open source projects only contain pre-trained generators and do not provide pre-trained discriminators. Although these pre-trained models are very high-quality generators, users cannot implement transfer learning in these generators. Because the training process of GAN is a confrontation between the generator and the discriminator. So if no suitable pre-trained model is found, this project will provides a complete pre-training model used in this experiment, so that other researchers can try their ideas on these models. Furthermore, due to the above reasons, the research results of GAN transfer learning are relatively less compared to other types of neural network model transfer learning, so many technologies are not perfect and there is no large-scale application that means this field still needs exploration. Although loading the parameters of the pre-trained model will improve the performance

of GAN, training GAN in a small data set is still a challenging task. First of all, the problem of overfitting needs to be solved when training GAN on small data sets. Then, although the existing transfer learning techniques can improve the performance of GAN, these methods still need to be improved.

3.2 Approach

This part mainly introduces the methods used in this project. First I will introduce the methods used in model pre-training. Next, through the analysis of the existing literature and research results, I think that combining the existing transfer learning techniques can maximize the advantages of each method. In addition, it is impossible for this project to use all the existing transfer learning techniques. In this section, I will briefly analyze why these methods can be used in this project and why other methods are excluded. Moreover, I will introduce the application of data augmentation to diversify the data when training GAN on a small data set. Finally, I will show how to evaluate the results of the experiment

3.2.1 Model pre-training

As mentioned in Chapter 2, choosing a suitable pre-training model is very important. Because different models will learn different prior knowledge, and will inevitably get different results of transfer learning. Obviously, these prior knowledge are the key factors for whether the pre-training model can efficiently complete the transfer learning. In addition, the diversity and density of the source domain data set is also one of the important factors affecting the effect of transfer learning. So the starting point of this project is to find a suitable pre-training model. I checked Pytorch's official model library Pytorch Hub (from: https://pytorch.org/hub/) and the famous code sharing platform github (from: https://github.com/), which contains a large number of free and open-source pre-training models. However, most pre-trained models only contain

generators, which means that these models cannot be used for GAN transfer learning. Since there is no suitable pre-training model, this project includes the pre-training model process and provides a complete pre-training model component (including discriminator and generator). In this section, I will introduce ideas on how to pre-train the model in this project. The original GAN will have problems such as mode collapse and unstable training. Therefore, many researchers have improved the original GAN from different aspects: DCGAN (improved on the network structure), WGAN/WGAn-GP (improved the loss function of the original GAN). Therefore, in order to improve the performance of the pre-training model, this project tests the loss function and training method of WGAN and WGAN-GP, and refers to the network structure of DCGAN. Next,i will introduce these GANs below.

DCGAN

In order to improve the original GAN network structure, Alec Radford & Luke Metz et al. proposed the use of deep convolution technology when designing the GAN network architecture (Alec and Luke 2016). This improvement makes the image quality generated by DCGAN much higher than the image quality generated by the original GAN (Alec and Luke 2016). In the paper, the author gives the following network structure that makes model training more stable (Alec and Luke 2016).

1. Use convolution and deconvolution instead of pooling layer
2. Add Batch normalization operations to both the generator and the discriminator.
3. Remove the fully connected layer.
4. The output layer of the generator uses Tanh activation function, and the other layers use RELU.
5. All layers of the discriminator use LeakyReLU activation function

In addition, in order to show the network structure more clearly, the paper gives the following picture (Alec and Luke 2016).

.

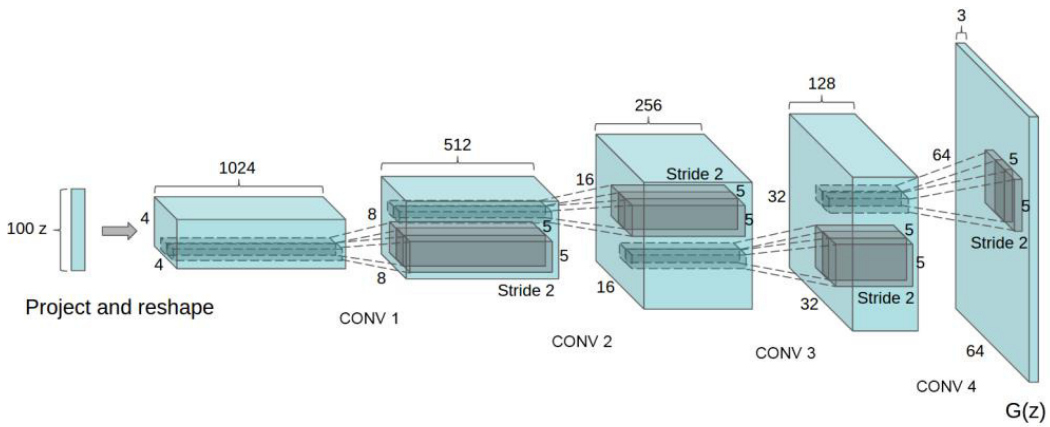


Figure 11: The network structure of DCGAN

For this project, I did not completely follow this architecture to build the generator and discriminator of the pre-trained network, I have just selected some of the improvements . First, the deconvolution technique is used in the generator, and the convolution layer is added to the discriminator. Then, use the batch normalization after the convolutional layer and the deconvolutional layer, which can speed up learning and convergence (transform the input of each layer to data with 0 mean and unit standard deviation). Moreover, it can solve the problem of difficulty in training due to improper parameter initialization. Finally,using the leaky relu activation function in the discriminator will improve the quality of the generated image.

WGAN/WGAN-GP

In order to improve the original GAN, Martin Arjovsky et al proposed WGAN in 2017. WGAN has the following advantages:Completely solve the problem of GAN training instability, no longer need to carefully balance the training level of the generator and the discriminator, solve the problem of model collapse and ensure the diversity of generated samples, no need for a well-designed network architecture, even if the generator and discriminator all use the fully connected layer,we can also achieved good results..

In addition, compared with the original GAN algorithm implementation process,

the improved WGAN has only four improvements (Martin et al. 2017):

1. Remove sigmoid in the last layer of the discriminator.
2. Optimize the loss function of generator and discriminator based on Wasserstein distance,so that they no longer take logarithms.
3. After each update of the parameters of the discriminator, truncate their absolute value to no more than a fixed constant , in order to make the parameters satisfy 1-Lipschitz.
4. Do not use momentum-based optimization algorithms (including momentum and Adam), the paper recommends RMSProp.

At the same time, the detailed training process of WGAN is given in the paper, as shown in the figure below (Martin et al. 2017).

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure 12: The training process of WGAN

On the other hand, some researchers believe that clipping the weight parameters in the neural network is too rough and causing the network to lose activity. So Ishaan Gulrajani et al improved the training strategy and loss function of WGAN. They introduced gradient penalty to make WGAN training more stable, and solved the problem of generating bad samples and unable to converge under special circumstances. Compared with WGAN, the main improvement of WGAN-GP is the loss function of the discriminator and the

authors gave the following formula (Ishaan et al. 2017).

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}\left[D(\tilde{\boldsymbol{x}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r}\left[D(\boldsymbol{x})\right]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

Figure 13: Loss function of the discriminator in WGAN-GP

According to the formula, the first part of the formula is the loss function of the WGAN, and the second part introduces a gradient penalty (Ishaan et al. 2017). λ is a custom hyperparameter, the authors suggest to set it to 10, and the optimizer of WGAN-GP is Adam (Ishaan et al. 2017). However, some people found in the project that images generated with WGAN are better than those generated with WGAN-GP. This project will also compare the performance of WGAN and WGAN-GP under the same network structure in the model pre-training stage.

3.2.2 Data set of pre-trained model

According to the description in Chapter 2, the data set used by the pre-training model is one of the important factors affecting the results of transfer learning.
For this case, I selected an Asian face data set during the model pre-training stage, because the human face has richer features, the pre-trained model can learn more prior knowledge.
The data set used for the pre-training model of this project is Asian face data set (from http://www.seeprettyface.com/mydataset_page3.html#wanghong). In addition, I will quote in the code according to the author's request. Please note that this data set cannot be used for commercial purposes according to the requirements of the data set author, and if you use this data set in your project, please cite it according to the author’s request.

3.2.3 Data augmentation

In the field of deep learning, the quantity and quality of data sets are important factors affecting model accuracy. For example, the model may overfit when the data is insufficient. For this case, data augmentation is often used in the field of image classification and image recognition to solve the problem of insufficient data sets. The main methods of data augmentation include image rotation, cropping, flipping, changing saturation and contrast, etc. Similarly, when the number of data sets is small, the pictures generated by GAN are blurry and the generated images are not diverse enough. So it is necessary to test different data augmentation methods to improve the quality and diversity of images generated by the generator in GAN. However, another problem that needs to be consider for this approach is that if the picture processed by the data augmentation technology is too different from the original picture, the ability of the discriminator to distinguish between real pictures and fake pictures will decrease, and it will also reduce the quality of the generated picture. For this situation, my solution is to slightly and randomly modify the image of the target domain during data augmentation and then i will compare the results of different data augmentation methods to select the methods suitable for this project.

3.2.4 Classify the layers of the pre-trained model

According to the above description, the main idea of this project is to combine the three existing transfer learning methods, including: network freezing, update the scale and shift, and model fine-tuning. Its purpose is to maximize the advantages of the three methods and make up for their shortcomings to a certain extent. The core of this method is to use the right technology for the right layer. I think that it is not comprehensive to only divide the pre-training model into the general part and the specific part , because freezing the middle

part may reduce the adaptability of the pre-training model to the target domain, and fine-tuning the middle part may lose prior knowledge and reduce training efficiency. So the pre-trained network needs to be classified:

1. The method used in the general part which learn low-level features is freezing layers.
2. The method used in the specific part which learn high-level features is model fine-tuning.
3. The method used in the middle part between the general part and the specific part is update the scale and shift.

Below I will briefly describe the transfer learning method used in this project.

3.2.4 Freeze general part

In transfer learning, fine-tuning the model directly in the target domain can improve training efficiency, but this method will make the model lose part of prior knowledge and increase the amount of parameter calculation. Because the model will learn some common knowledge of multiple domains during the pre-training process and the main purpose of transfer learning is to retain this information. In most cases, the previous convolutional layer will learn features which are suitable for several domains, and the following fully connected layer will learn specific knowledge of a certain domain. Therefore, a more effective method in transfer learning is to freeze the convolutional layer and then re-initialize the fully connected layer, it can reduce the number of parameters that need to be optimized in transfer learning. But in the Generative Adversarial Network (GAN), the existence of generators needs to be considered, the literature review discussed which layers of the generator and discriminator can be used as a general part of GAN. In addition, there is a paper that specifically describes the advantages of freezing the discriminator in the transfer learning of GANs (SangwooMo et al. 2020). However, in different network structures, the layers for learning these general features are

also different. Overview, the main purpose of this method is to find the general parts in the model and freeze them which is an important work in the transfer learning of GANs.

3.2.5 Update the scale and shift

As mentioned in Chapter 2, There are two existing methods for updating scale and shift, Filter Selection (FS) and Adaptive Filter Modulation (AdaFM). Compared with FS, AdaFM has higher adaptability to the target domain. Therefore, this project will use the AdaFM for the middle part of the model. Next, I will briefly introduce this method. During the training process, the parameters of the convolution kernel are fixed, and the trainable parameter scale (representing the activation strength of the convolution filter) and the shift (equivalent to the activation threshold of each convolution kernel) are added to the outside of the convolution kernel (Miaoyun et al. 2020). Although this method improves the adaptability to the target domain compared to the frozen convolution kernel, it also increases the amount of parameter calculation. Therefore, the combination of this method and the freezing network is not only to improve the adaptability of the pre-training model to the target domain, but also to reduce the amount of parameter calculation to improve training efficiency.

In addition, in the paper mentioning FS, the authors also introduced another method, which is to use this method in a pre-trained generator (GAN or other generative network), and use VGG16 to do work similar to a discriminator. Although the authors modified the loss function, I think this method has some problems:

1. If the depth difference between the pre-trained generator and VGG16 is too large, the two models may not match each other.
2. When the size of the picture generated by the pre-trained generator is different from the input size of VGG16, the picture needs to be up-sampled or

down-sampled, and some features may be lost.

3. If the target domain is not in the VGG16 pre-trained data set, VGG16 may not be able to accurately extract the features of the target domain.

For these points, I think this method is not suitable for this project.

3.2.6 Fine-tune specific part

Although the transfer learning method is continuously improved, model fine-tuning is still a simple and effective method. Especially when the neural network model learns the high-level features of the target domain. Obviously, for the unique features of the target domain, the pre-training model does not acquire this knowledge from the source domain. Therefore, they need to be relearned in the process of transfer learning. On the other hand, fine-tuning the entire pre-training model will waste too much time and computer resources. For this case, it is necessary to decompose the pre-trained network to find the layers that have learned high-level features. Therefore, only the specific part of the model are fine-tuned in this project.

3.2.8 Use FID to evaluate results

In order to ensure that each layer in the network can use appropriate methods to implement transfer learning, this project will use a metric to evaluate different combinations. For GANs, the most commonly used evaluation metrics are IS and FID. Inception Score (IS) uses an image classifier pre-trained on ImageNet (Qiantong et al. 2018). That means if the data set is not in ImageNet, IS cannot accurately assess the quality of GAN generated pictures. In addition, IS cannot detect the problem of mode collapse. Although The Fréchet Inception Distance (FID) also uses the same pre-trained model as IS to extract the features of the generated images, it also extracts the features of the images in the dataset . Then use the Fréchet distance to evaluate the distance between the two distributions (Qiantong et al. 2018). Therefore, FID is more

widely used and has more reference value. In addition, since the animation face dataset used in this project is not included in ImageNet, I decided to use FID to evaluate the experimental results.

Chapter 4 Products

Recently, as deep learning has become a hot field, there are more and more frameworks for deep learning, including Tensorflow, Pytorch, Caffe, Keras, Theano, etc. Before starting to experiment, my priority should be which deep learning framework to choose to implement my ideas. Obviously, the most popular deep learning framework is Tensorflow, but it is in the phase of alternating between Tensorflow1.0 and Tensorflow2.0. If you use Tensorflow1.0, some code will not run in the future. On the other hand, Tensorflow2.0 may also have some bugs due to its recent launch, and it has no large-scale applications. So I think Pytorch is the suitable tool for this project. In addition, Pytorch also supports dynamic graph operations that means it can improved coding efficiency (All codes of this project are running in Pytorch 1.6.0). This chapter mainly introduces requirements, codes and results.

4.1 Requirement

4.1.1 Implement model pre-training

As mentioned above, the pre-training model is one of the important factors that determine the results of transfer learning. Therefore, the first requirement in the implementation phase of this project is to design a suitable network structure, and select a model with higher performance from WGAN and WGAN-GP as the pre-training model.

4.1.2 Diversify the data of the target domain

As described in Chapter 3, the problem to be considered in this method is to diversify the data of the target domain under the premise of ensuring the performance of the GAN. Therefore, the main requirement of this part is to find a data augmentation method suitable for GAN. In addition, the same method given different values will also get different results, such as the picture rotated by 90 degrees and 180 degrees. So finding the right parameters is also one of the requirements of this part.

4.1.3 Use different methods for different parts of the model

The requirement of this part is to load the parameters of the pre-trained model and then use different transfer learning methods for different parts of the model, including freezing the network, updating the scale and shift, and fine-tuning. Especially for updating scale and shift, it needs to add two learnable parameters to the convolution kernel that means I need to add some custom operations.

4.2 Design

This part will introduce the design decision of this project, the factors considered in the experiment and the estimation of the result based on the existing conclusion.

1. In the experiment, I will compare different methods at each stage. For example, I will compare the results of WGAN and WGAN-GP in the model pre-training phase. In order to improve the efficiency of the experiment, I will compare their results after 8000 training iterations, and then select the model with better results to continue training.

2. As mentioned in Chapter 3, the scale of data augmentation needs to be limited. If the processed image is too different from the original image, this may

reduce the quality of the image generated by the model. In addition, these parameters need to be adjusted in following experiments.

3. For the GAN network structure, both the generator and the discriminator have certain symmetry. For example, the generator and the discriminator have the same number of layers, the first layer of the discriminator is the convolutional layer and the last layer of the generator is the deconvolution layer. In addition, I predict that this symmetry not only appears in the design of the GAN network structure, but also applies to the classification of pre-trained networks in transfer learning. For example, if the first layer of the discriminator is the general part, the last layer of the layer generator has a high probability of being the general part.

4. For the evaluation of experimental results, the most common way to evaluate GAN is IS score or FID score. These scores need to use the Inception model pre-trained on the ImageNet (Qiantong et al. 2018). However, the target domain of this project uses anime face dataset, which is not included in ImageNet. Although FID is used to evaluate the results in this project, its reference value may decrease.

4.3 Implementation and results

4.3.1 Model pre-training

In this part, my main task is to design the network architecture of the pre-training model according to the method mentioned in Chapter 3 and implement it in Pytorch. The following pictures respectively show the network structure of the generator and discriminator. For this network structure, I used part of the design ideas of DCGAN.

(The network structure and weight initialization refer to https://github.com/pytorch/examples/tree/master/dcgan)

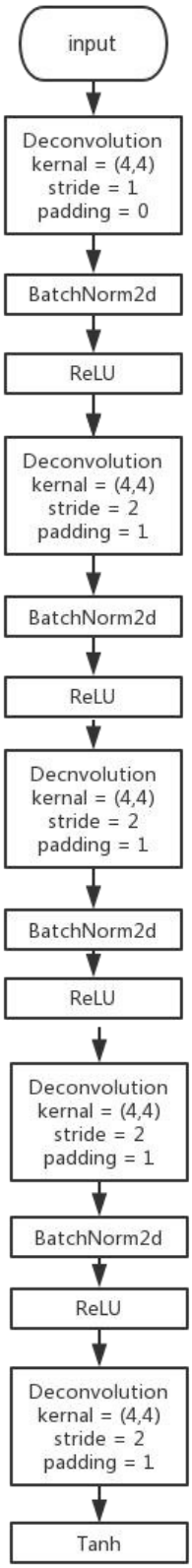


Figure 14: Generator network structure

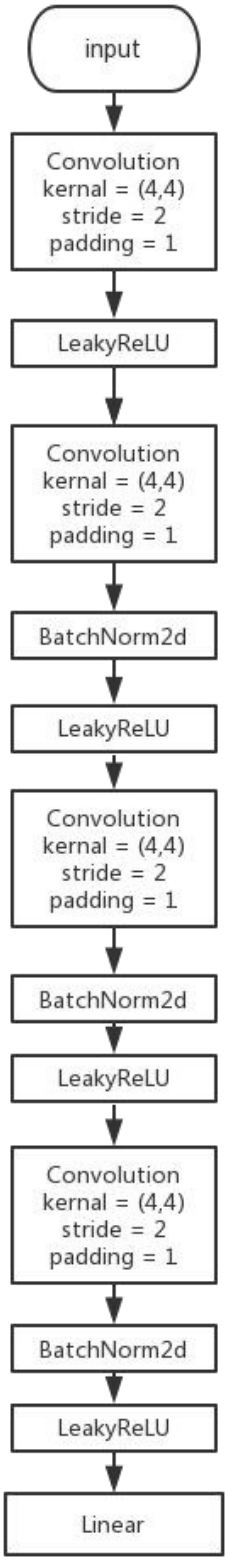


Figure 15: The network structure of the discriminator

In this project, I will use the same network structure to test the performance of WGAN and WGAN-GP. Finally, both WGAN and WGAN-GP optimize the discriminator five times first, and then optimize the generator once. Moreover, in order to prevent unexpected problems such as system interruption, hardware failure, I save the model every one thousand iterations.

As mentioned in chapter 3, I selected 20,000 images in an Asian face dataset , although the GAN that generates faces is relatively difficult to train. Before I started training, I found that there are many pictures in the data set with different expressions on the same face. So in order to get better training results, I shuffle the data set and select 20,000 pictures for training (data set from http://www.seeprettyface.com/ mydataset_page3.html#wanghong).

The pre-training process takes a very long time. Therefore, in order to improve the efficiency of the experiment, I will compare the results of WGAN and WGAN-GP in the first 8000 training iterations. In addition, GAN has uncertainty. For example, the pictures generated by the 3000th iteration may be better than the pictures generated by the 5000th iteration. For this case, I will compare the results every 1000 iterations. Finally, I will choose a model with a higher quality of generated pictures as the pre-training model for this project, and increase the number of iterations. The following is a comparison of WGAN-GP and WGAN generated pictures.

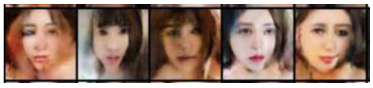
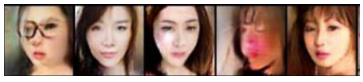
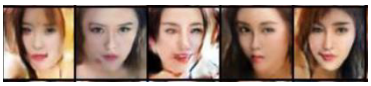
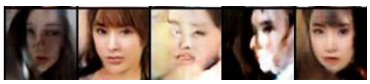
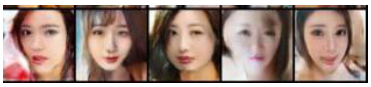
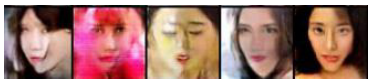
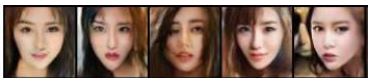
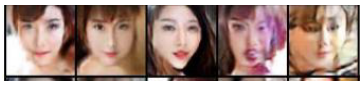
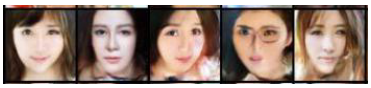
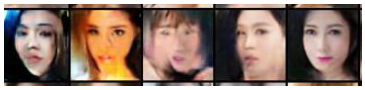
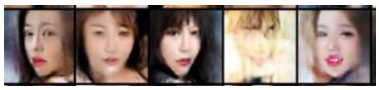
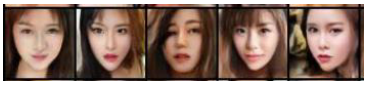
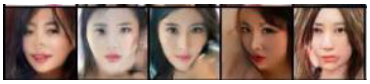
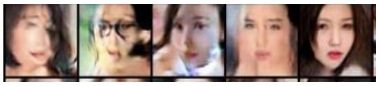
Epoch	WGAN	WGAN-GP
0		
1000		
2000		
3000		
4000		
5000		
6000		
7000		
8000		

Table 1: Comparison of WGAN and WGAN-GP (from the 0th iteration to the 8000th iteration, the two sets of pictures are compared once every thousand times)

The above chart shows that the performance of WGAN is higher than the performance of WGAN-GP. However, this result can not prove that the image quality generated by WGAN must be higher thanthe image quality by WGAN-GP. Just in some network architectures the performance of WGAN is relatively better. In addition, the data set, the number of iterations and other factors will also affect the results of GAN. Due to the relatively high quality of

images generated by WGAN, this project will use WGAN as a pre-training model. (Training 20,000 times, training time is five days, training equipment is NVIDIA GeForce RTX 2080Ti). This also proves that training GAN takes muchtime and computer resources. Furthermore, when using non-public data sets for commercial purposes, the cost of data is also very large, such as data collection, data cleaning, etc. These issues also illustrate the importance of developing GAN transfer learning technology.

4.3.2 Data augmentation

As mentioned in the model pre-training, training a GAN requires a large data set, which means an increase in time and computer resource costs. Therefore, in order to solve this problem, the goal of this project is to train GAN in a small dataset, using pre-trained models. However, one problem that cannot be ignored for small data sets is overfitting. For this situation, I applied data augmentation technology to this project. The main tool is the torchvision.transforms module in the Pytorch library, which can achieve a combination of multiple data augmentation technologies. Below I will show the code to implement data augmentation, and the picture after using data augmentation technology. The main task of this part is to increase the diversity of data and retain the features of the original picture.

(The data set of the target domain is a cartoon face dataset and I chose 96 images. The author of this dataset is the same as the author of the source domain dataset, so please check the description of the dataset in Chapter 3 or the website before using it.
Data set form http://www.seeprettyface.com/mydataset.html)

```python
def loadtraindata():
    path = r"D:\WGAN\train_images\target_domain"
    train_data = torchvision.datasets.ImageFolder(path,transform=transforms.Compose([
                transforms.ColorJitter(brightness=0.3,contrast=0.2),transforms.RandomRotation(5),transforms.CenterCrop(510),
                transforms.Resize((64,64)),transforms.ToTensor(), transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5))]))
    train_loader = torch.utils.data.DataLoader(train_data, batch_size= 96,drop_last = True,shuffle = False)
    return train_loader
```

Figure 16: The code of data augmentation

Original target domain dataset	Data augmentation 1	Data augmentation 2

Table 2: Comparison of the data augmentation and the original image

4.3.3 Load pre-trained model

Before implementing GAN transfer learning, I need to load the parameters of the pre-trained model. For this requirement, Pytorch provides a very convenient function: model.load_state_dict(torch.load('model path')). However, this function does not make too many modifications to the pre-trained network, such as convolution kernel operations, etc. Therefore, I used the torch.load('model path') function to load the pre-trained model. This method can selectively load pre-trained parameters, and can control which parameters are learnable and which parameters need to be fixed. Next, And when I built the convolutional layer of the network, I did not use the commonly used the nn.Sequential(nn.Conv2d()) function in Pytorch, but used the torch.nn.functional.conv2d() function. Because this function can customize the initialization of the convolution kernel, this feature is very convenient for adding operations to the convolution kernel. The following pictures are the code to

load the pre-trained network.

```
class Generator_transfer(nn.Module):

    def __init__(self):
        super(Generator_transfer, self).__init__()
        self.pre_train_model = torch.load('D:\WGAN\models\wgangp_wanghong_g_epoch20000.pth')
        self.convt_keys, self.bn_keys = self.get_pre_train_weight()
        self.dimz = 100
        self.gc = 64  # generate channel
        self.conv1_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[0]], requires_grad=False)
        self.conv2_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[1]], requires_grad=False)
        self.conv3_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[2]], requires_grad=False)
        self.conv4_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[3]], requires_grad=False)
        self.conv5_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[4]], requires_grad=False)

        self.bn1 = nn.Sequential(nn.BatchNorm2d(self.gc * 8))
        self.bn2 = nn.Sequential(nn.BatchNorm2d(self.gc * 4))
        self.bn3 = nn.Sequential(nn.BatchNorm2d(self.gc * 3))
        self.bn4 = nn.Sequential(nn.BatchNorm2d(self.gc ))

        self.bn1[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[0]], requires_grad=False)
        self.bn1[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[1]], requires_grad=False)
        self.bn1[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[2]], requires_grad=False)
        self.bn1[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[3]], requires_grad=False)

        self.bn2[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[4]], requires_grad=False)
        self.bn2[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[5]], requires_grad=False)
        self.bn2[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[6]], requires_grad=False)
        self.bn2[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[7]], requires_grad=False)

        self.bn3[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[8]], requires_grad=False)
        self.bn3[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[9]], requires_grad=False)
        self.bn3[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[10]], requires_grad=False)
        self.bn3[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[11]], requires_grad=False)

        self.bn4[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[12]], requires_grad=False)
        self.bn4[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[13]], requires_grad=False)
        self.bn4[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[14]], requires_grad=False)
        self.bn4[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[15]], requires_grad=False)
```

Figure 17: Load the parameters of the pre-trained generator

```
def forward(self, input):

    output = F.conv_transpose2d(input,weight=self.conv1_kernel,stride = 1,padding =0)
    output = self.bn1(output)
    output = F.relu(output,inplace=True)
    output = F.conv_transpose2d(output,weight=self.conv2_kernel,stride = 2,padding =1)
    output = self.bn2(output)
    output = F.relu(output,inplace=True)
    output = F.conv_transpose2d(output,weight=self.conv3_kernel,stride = 2,padding =1)
    output = self.bn3(output)
    output = F.relu(output,inplace=True)
    output = F.conv_transpose2d(output,weight=self.conv4_kernel,stride = 2,padding =1)
    output = self.bn4(output)
    output = F.relu(output,inplace=True)
    output = F.conv_transpose2d(output,weight=self.conv5_kernel,stride = 2,padding =1)
    output = torch.tanh(output)
    return output
```

Figure 18: Rewrite the forward function of the generator

```python
def get_pre_train_weight(self):
    convt_keys = ["net.0.weight","net.3.weight","net.6.weight","net.9.weight","net.12.weight"]
    bn_keys = ["net.1.running_mean","net.1.running_var","net.1.weight","net.1.bias",
               "net.4.running_mean","net.4.running_var","net.4.weight","net.4.bias",
               "net.7.running_mean","net.7.running_var","net.7.weight","net.7.bias",
               "net.10.running_mean","net.10.running_var","net.10.weight","net.10.bias"]
    return convt_keys,bn_keys
```

Figure 19: Get the key of parameters in generator

```python
class Discriminator_transfer(nn.Module):

    def __init__(self):
        super(Discriminator_transfer, self).__init__()
        self.dc = 64  # discriminative channel
        self.pre_train_model = torch.load('D:\WGAN\models\wgangp_wanghong_d_epoch20000.pth')
        self.convt_keys, self.bn_keys = self.get_pre_train_weight()

        self.conv1_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[0]], requires_grad=False)
        self.conv2_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[1]], requires_grad=False)
        self.conv3_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[2]], requires_grad=False)
        self.conv4_kernel = nn.Parameter(data=self.pre_train_model[self.convt_keys[3]], requires_grad=False)

        self.bn1 = nn.Sequential(nn.BatchNorm2d(self.dc * 2))
        self.bn2 = nn.Sequential(nn.BatchNorm2d(self.dc * 4))
        self.bn3 = nn.Sequential(nn.BatchNorm2d(self.dc * 8))

        self.bn1[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[0]], requires_grad=False)
        self.bn1[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[1]], requires_grad=False)
        self.bn1[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[2]], requires_grad=False)
        self.bn1[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[3]], requires_grad=False)

        self.bn2[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[4]], requires_grad=False)
        self.bn2[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[5]], requires_grad=False)
        self.bn2[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[6]], requires_grad=False)
        self.bn2[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[7]], requires_grad=False)

        self.bn3[0].running_mean = nn.Parameter(data=self.pre_train_model[self.bn_keys[8]], requires_grad=False)
        self.bn3[0].running_var = nn.Parameter(data=self.pre_train_model[self.bn_keys[9]], requires_grad=False)
        self.bn3[0].weight = nn.Parameter(data=self.pre_train_model[self.bn_keys[10]], requires_grad=False)
        self.bn3[0].bias = nn.Parameter(data=self.pre_train_model[self.bn_keys[11]], requires_grad=False)

        self.fc =  nn.Sequential(nn.Linear(8192,1))
```

Figure 20: Load the parameters of the pre-trained discriminator

```python
def forward(self, input):
    output = F.conv2d(input,weight=self.conv1_kernel,stride=2,padding=1)
    output = F.leaky_relu(output,0.2,inplace=True)
    output = F.conv2d(output,weight=self.conv2_kernel,stride=2,padding=1)
    output = self.bn1(output)
    output = F.leaky_relu(output,0.2,inplace=True)
    output = F.conv2d(output,weight=self.conv3_kernel,stride=2,padding=1)
    output = self.bn2(output)
    output = F.leaky_relu(output,0.2,inplace=True)
    output = F.conv2d(output,weight=self.conv4_kernel,stride=2,padding=1)
    output = self.bn3(output)
    output = F.leaky_relu(output,0.2,inplace=True)
    output = output.view([input.size()[0],-1])
    output = self.fc(output)
    return output
```

Figure 21: Rewrite the forward function of the discriminator

```python
def get_pre_train_weight(self):
    convt_keys = ["net.0.weight","net.2.weight","net.5.weight","net.8.weight"]
    bn_keys = ["net.3.running_mean","net.3.running_var","net.3.weight","net.3.bias",
               "net.6.running_mean","net.6.running_var","net.6.weight","net.6.bias",
               "net.9.running_mean","net.9.running_var","net.9.weight","net.9.bias"]
    return convt_keys,bn_keys
```

Figure 22: Get the key of parameters in discriminator

4.3.4 Model fine-tuning

According to the pre-trained network structure ,the discriminator consists of four convolutional layers and one fully connected layer, and the generator consists of five deconvolutional layers. First of all, compared with the general part and the middle part, the specific part is easier to find. The approach to find the specific part is to gradually freeze the layers except the last layer in the discriminator and the layers in the generator except the first layer. Then I will fine tune the remaining layers. If the results is similar to the source domain and very different from the target domain, that means I need to fine-tune more layers. Next, I will compare the results of freezing different layers.

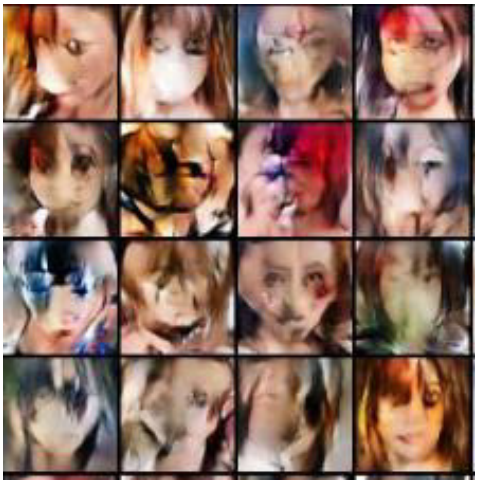


Figure 23: Freeze four layers in G and D respectively.

The above figure is the result of freezing the first four layers of the discriminator and the last layer of the generator, and then fine-tuning the last

layer of the discriminator and the first layer of the generator. The results show that the generated images are closer to the source domain that means they retain most of the features in the source domain. Therefore, except the fully connected layer of the discriminator and the first deconvolution layer of the generator, more layers also need to be fine-tuned. In the following experiment, I respectively freeze three layers in the generator and discriminator and then get the following result.

Figure 24: Freeze three layers in G and D respectively.

The above figure shows that the result of fine-tuning the two layers is obviously closer to the target domain. For this result, I found that the specific part of the model is the first two layers of the generator and the last two layers of the discriminator. However, the adaptability of the model to the target domain still needs to be improved. In general, although it was pointed out in some papers that the convolutional layer of the model learned common features applicable to multiple domains, I found in experiments that some convolutional layers learned specific features of a certain domain. Therefore, some convolutional layers also need to be fine-tuned.

4.3.5 Frozen layers ,updating scale and shift

The main purpose of this part is to analyze the remaining layers in the network. First of all, for the discriminator, from the upper network to the lower network become more and more specific. On the contrary, for the generator, the upper network to the lower network become more and more general. So the last layer of the generator and the first layer of the discriminator must be the general part of the model. For the remaining layers, which layers are frozen and which layers update scale and shift can get the best results, that need to be verified through experiments. According to the formula for AdaFM, when adding scale and shift parameters to the convolution kernel, each convolution kernel has the same scale and shift. For example, a 3*3 convolution kernel will multiply the same scale and add the same shift. In addition, in the initial state, the initial value of scale is 1, and the initial value of shift is 0. Therefore, this project should be close to their initial values when initializing these parameters. Below is the code for initializing these parameters and comparison with different combinations.

```
self.conv3_scale = nn.Parameter(data=torch.FloatTensor(256, 128, 1, 1).normal_(1.0, 0.1), requires_grad=True)
self.conv3_shift = nn.Parameter(data=torch.FloatTensor(256, 128, 1, 1).normal_(0.001, 0.001), requires_grad=True)
self.conv4_scale = nn.Parameter(data=torch.FloatTensor(128, 64, 1, 1).normal_(1.0, 0.1), requires_grad=True)
self.conv4_shift = nn.Parameter(data=torch.FloatTensor(128, 64, 1, 1).normal_(0.001, 0.001), requires_grad=True)
```

Figure 25: Code to initialize scale and shift

In order to show the combinations which are used in the experiment more clearly, I will describe them with numbers, include G(i,j,k) and D(i,j,k).

1. For D(i,j,k), the i means to freeze the first i layers in the network, the j means to update the scale and shift from the i layer to the i+j layer, and the k means to fine-tune the remaining k layers. For example, D(2,1,2) equivalent to freezing the first two layers of discriminator, updating the scale and shift in the third layer, and fine-tuning the fourth and fifth layers.

2. For G(i,j,k), the k means to fine-tune the first k layers in the network, the j means to update the scale and shift from the k layer to the k+j layer, and the i means to freeze the remaining i layers. For example, G(1,2,2) equivalent to freezing the fifth layer of generator, updating the scale and shift from the third layer to the fourth layer, and fine-tuning the first two layers.

Below I will show the experimental results of different combinations. Moreover, the following experimental results may not be very obvious, so I use FID to evaluate these results more accurately. The code of FID from https://github.com/mseitzer/pytorch-fid (Maximilian 2020), I also cite it in the code according to the author's requirement.

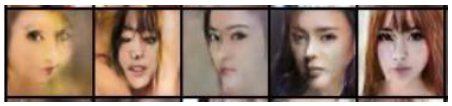
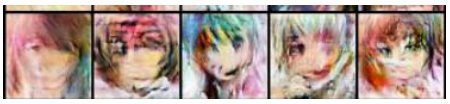
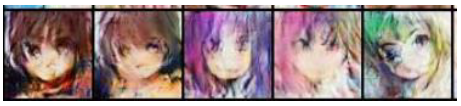
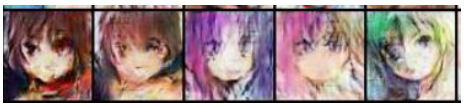
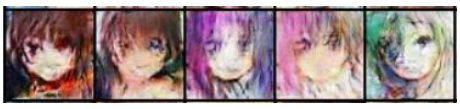
Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 3: The results of D(2,1,2), G(2,1,2)

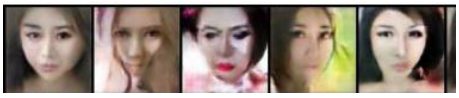
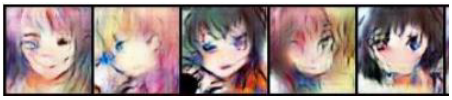
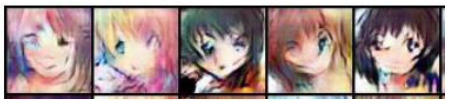
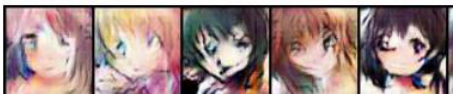
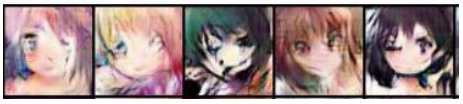
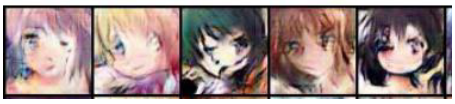
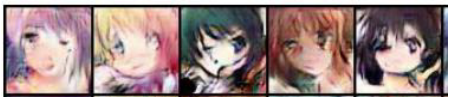
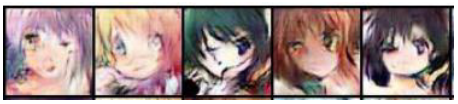
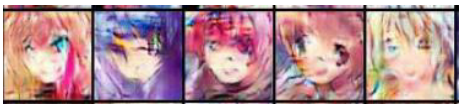
Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 4: The results of D(1,2,2), G(1,2,2)

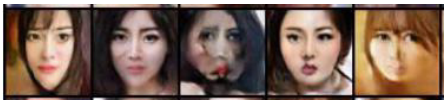
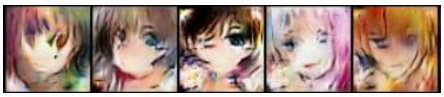
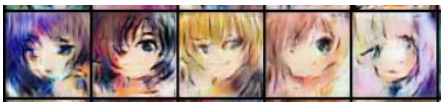
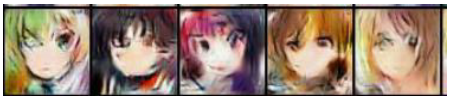
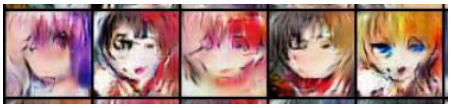
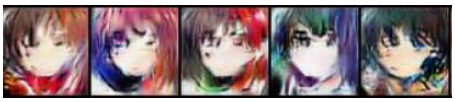
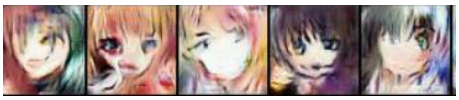
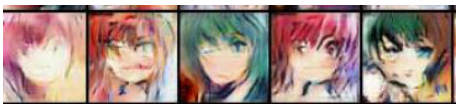
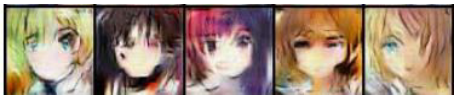
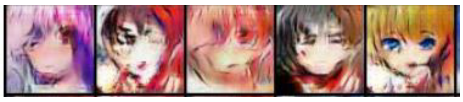
Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 5: The results of D(1,2,2), G(3,0,2)

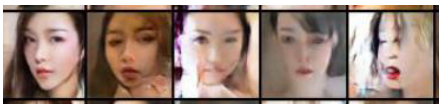
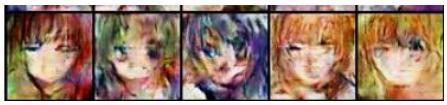
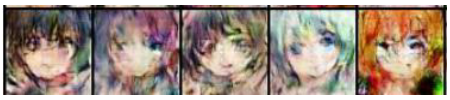
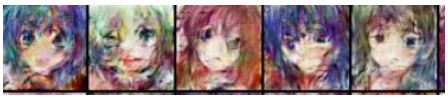
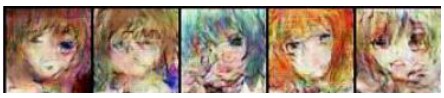
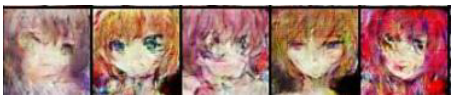
Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 6: The results of D(3,0,2), G(1,2,2)

Combination	FID
D(2,1,2), G(2,1,2)	220
D(1,2,2), G(1,2,2)	**195**
D(1,2,2), G(3,0,2)	204
D(3,0,2), G(1,2,2)	223

Table 7: FID of these combinations .

According to the results, D(1,2,2), G(1,2,2) is the best combination of transfer learning methods.

4.3.6 Compare with other methods

In this part, the transfer learning methods used in this project will be tested individually and then compared with the combination of these methods. In order to eliminate the influence of other factors, I will use the same data set and training iterations as the above experiment.

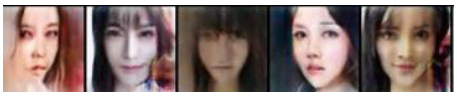
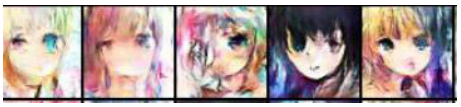
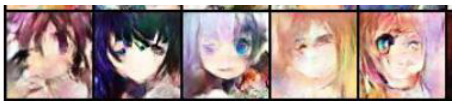
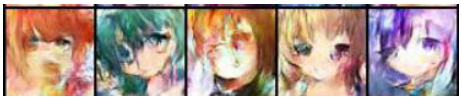
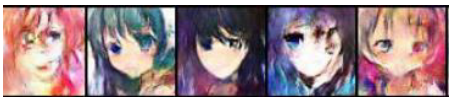
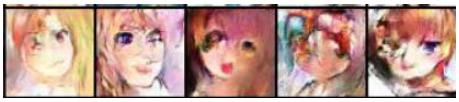
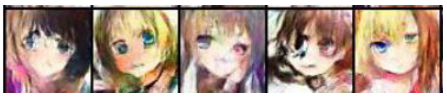
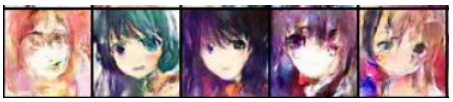
Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 8: The result of model fine-tuning

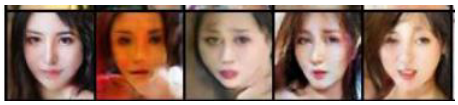
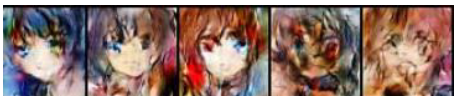
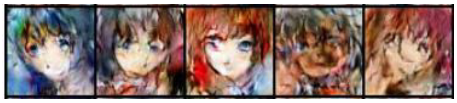
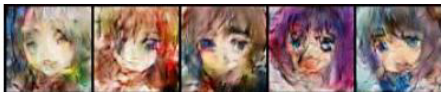
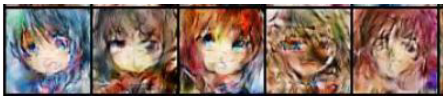
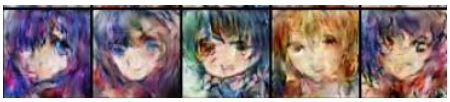
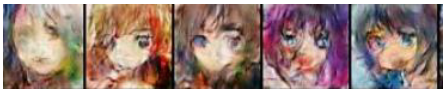
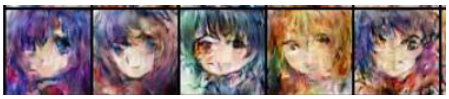
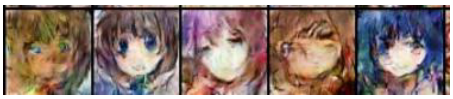
Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 9: Freeze three layers of G and D respectively

Epoch	Result
0	
2000	
4000	
6000	
8000	
10000	
12000	
14000	
16000	
18000	
20000	

Table 10: Update scale and shift

Transfer learning method	FID
Model fine-tuning	209
Frozen layers	225
Update scale and shift	236
Combination-D(1,2,2), G(1,2,2)	**195**

Table 11: Compare FID of combination and other methods

The above experimental results show that the combination of existing transfer learning techniques can improve the performance of the model. Moreover, the results obtained by combining methods are better than using these transfer learning techniques alone. In addition, according to the experimental results, if AdaFM is added to the general part of the model, the quality of the pictures generated by the model will be reduced.

Chapter 5 Analysis

This chapter will include some analysis based on the results of the experiment and the description of the success of the experiment. Then summarize the limitations of this project

5.1 Analysis of model pre-training

In the model pre-training phase, the quality of the pictures generated by WGAN-GP is not better than that of WGAN, which proves that the performance of WGAN-GP is not higher than that of WGAN under any circumstances. Network structure and data set are also important factors affecting model performance. Furthermore, model pre-training takes much time and computer resources, and training models from scratch requires much more data than transfer learning.

5.2 Analysis of data augmentation

For small data sets, the problem of overfitting cannot be ignored, so it is necessary to use some data augmentation techniques before model training. At the same time, it is found that the image enhancement methods need to be changed according to the use scenarios, and the amount of new data generated needs to be controlled within a certain range, in order to reduce the adverse impact on the generated results.

5.3 Analyze the combination of methods

First of all, it is relatively simple to locate a specific part of the network, because its results are more obvious. For example, if the layers of a specific part are frozen, the image generated by the generator will be very close to the source domain and very different from the target domain. For the remaining

layers, users need to try different combinations and compare their results. For classification networks, users can try to classify based on the symmetry of the generator and the discriminator. For example, the first layer of the discriminator is the general part, and the last layer of the generator is also likely to be the general part. According to the above experimental results, for the WGAN architecture used in this experiment, the best combination of transfer learning techniques is G(1,2,2),D(1,2,2).

5.4 Achieved goals and needs

At first, this paper proves that in the neural network, not only the fully connected layer learns specific knowledge of a certain domain, but some convolutional layers near the fully connected layer also learn these unique knowledge. In transfer learning, if you freeze these layers, you may not get the desired results. Then this experiment uses three GAN transfer learning methods, including model fine-tuning, freezing layers, updating scale and shift. These methods can effectively transfer the prior knowledge learned by the pre-trained GAN to the target domain. But these methods also have some shortcomings. For example, model fine-tuning will increase the amount of parameter calculation, and freezing the network will reduce the adaptability of the pre-training model to the target domain. In order to solve these problems, this project combines these methods and proves that this combination can make the model generate higher quality pictures. In addition, this project also proposed that when GAN is trained on a small data set, data augmentation should be used to reduce the risk of overfitting.

5.5 Limitations

Firstly, this project proposes a combination of transfer learning methods, but the combination method used in this project is not suitable for all situations. Because different network structures, different training strategies and different

data sets are important factors that need to be considered. Therefore, in actual use, users need to modify the combination method proposed in the paper according to the model and data set they use. Furthermore, due to time and computer resource constraints, I did not try these transfer learning methods on GANs that can generate high-resolution images (like: stylegan or biggan) in this project. Next, this project proposes a relatively fast method of classifying networks which compares the quality of the images generated by the model in the early and mid-term. However, this method may also have some limitations in some special cases. For example, the difference between the generated pictures is not obvious or the IS or FID score cannot be used to evaluate the result. These limitations may reduce the efficiency of transfer learning in some special cases.

Chapter 6 Conclusions and future work

This chapter will first introduce some conclusions based on experimental results in this project. Then I will show my plans for future work.

6.1 Conclusions

In general, through the experiments in this project, I got the following conclusions. First of all, compared to training the model from scratch, using a pre-trained GAN on the target domain will greatly improve the performance of the model, including speeding up training, reducing data requirements and improving the quality of generated pictures. Furthermore, in commercial use, the advantages of GAN's transfer learning are more obvious. Because in most cases public data sets are not allowed for commercial use, this further increases the cost of collecting data. If some companies use a large number of data to train GAN from scratch, more resources will be wasted. Then for WGAN and WGAN-GP, WGAN-GP is not better than WGAN at all times. For example, in this experiment, WGAN can generate higher quality pictures compared to WGAN-GP. Although some GAN researchers will compare their results with other GANs to prove their success, these results are not applicable to all situations. Obviously, the results of GAN are affected by many factors, such as network structure, number of iterations, and data sets. Next, using a combination of different transfer learning technologies can maximize the advantages of these technologies. For example, speed up the convergence speed of the pre-training model in the target domain, and improve the adaptability of the pre-training model to the target domain, etc. In addition, this combination will make up for the shortcomings of these methods to a certain extent. Compared with using these transfer learning methods alone, combining these methods can get better results. Finally, for small data sets, using some data augmentation techniques will reduce the risk of overfitting. However,

when using these technologies, the quality of the generated pictures may be reduced. Therefore, different data augmentation techniques should be selected for different data sets.

6.2 Future work

In this section, I will introduce some optional future work. As mentioned in the analysis, this project still has some limitations. For example, different combinations need to be used for different GANs. How to improve the versatility of this combination scheme is a necessary work. In this case, I will try this combination in more GANs in future work, and use different data sets for testing. In addition, how to generate images of multiple target domains in a pre-training model is also a very meaningful direction, because this method can further improve training efficiency. For this requirement, I think we can refer to the related technologies of conditional GAN, such as labeling different target domain data and passing them to the discriminator, turning unsupervised learning into supervised learning. Finally, exploratory development of GAN transfer learning technology is also one of the directions of future work. For example, optimize the network structure of the pre-training model, modify the loss function during transfer learning, etc. This can not only improve the existing GANs transfer learning method, but also provide more options for the combination of GAN transfer learning methods.

Chapter 7 Reflection

This part mainly introduces the reflections I summarized in this project. The first one is to fully understand the background knowledge of the project before starting a project. For example, at the beginning of this project, I only understood the principles and training process of GAN, but did not read the literature on transfer learning, which made me choose a wrong research direction when i started the project. Then it is necessary to read the existing reference materials with critical thinking and dare to discover their problems. As described in the conclusion, many studies only show their success, but do not describe their limitations. This requires readers to analyze based on existing knowledge. Finally, all ideas need to be verified through experiments. Because the results of the experiment can help me find some problems in my ideas and correct them in time.

Reference list:

[1] Yaxing,W. et al. 2018. Transferring GANs: generating images from limited data *ECCV*, pp. 2-17.

[2] Ya¨el,F. and Jean-Baptiste,G. 2019. *Mind2Mind : transfer learning for GANs* [Preprint]. Available at: http://arxiv.org/pdf/1906.11613v1.pdf [Accessed: 27 Jun 2019].

[3] Atsuhiro,N. and Tatsuya,H. 2019. *Generation From Small Datasets via Batch Statistics Adaptation*[Preprint]. Available at: http://arxiv.org/pdf/1904.01774v4.pdf [Accessed: 23 Oct 2019].

[4] SangwooMo,K. et al. 2020. *Freeze the Discriminator: a Simple Baseline for Fine-Tuning GANs*[Preprint]. Available at: http://arxiv.org/pdf/2002.10964v2.pdf [Accessed: 28 Feb 2020].

[5] Goodfellow,LanJ. et al. 2014. *Generative Adversarial Nets*[Preprint]. Available at: http://arxiv.org/pdf/1406.2661v1.pdf [Accessed: 10 Jun 2014].

[6] Martin,A. et al. 2017. *Wasserstein GAN*[Preprint]. Available at: http://arxiv.org/pdf/1701.07875v3.pdf [Accessed: 6 Dec 2017].

[7] Ishaan,G. et al. 2017. *Improved Training of Wasserstein GANs*[Preprint]. Available at: http://arxiv.org/pdf/1704.00028v3.pdf [Accessed: 25 Dec 2017].

[8] Wei,Z. 2017. Research on the deep learning of the small sample data based on transfer learning. *AIP Conference Proceedings 1864*(020018), pp. 2-8. doi: https://doi.org/10.1063/1.4992835

[9] Alec,R. and Luke,M. 2016. *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS*[Preprint]. Available at: http://arxiv.org/pdf/1511.06434v2.pdf [Accessed: 7 Jan 2016].

[10] Qiantong,X. et al. 2018. *An empirical study on evaluation metrics of generative adversarial networks* [Preprint]. Available at: http://arxiv.org/pdf/1806.07755v2.pdf [Accessed: 17 Aug 2018].

[11] Miaoyun,Z. et al. 2020. *On Leveraging Pretrained GANs for Generation with Limited Data* [Preprint]. Available at: http://arxiv.org/pdf/2002.11810v3.pdf [Accessed: 8 Aug 2020].

[12] Gwylab. 2019. Download and introduction of the data set Available at: http://www.seeprettyface.com/mydataset.html [Accessed: 06 March 2019].

[13] Maximilian,S. 2020. *pytorch-fid: FID Score for PyTorch.* Version 0.1.1 [Source Code]. Available at: https://github.com/mseitzer/pytorch-fid [Accessed: 3 Dec 2020]

[14] Pytorch. 2020. *DCGAN* [Source Code]. Available at: https://github.com/pytorch/examples/tree/master/dcgan [Accessed: 06 Jul 2020]

Appendices

Files and folders	Description
fid	Calculate the FID of GAN(Maximilian 2020)
generated_image	The folder for temporarily storing the generated pictures, the pictures in the folder are used to calculate the FID
pretrained-model	Folder where pre-trained models are stored
source_domain	Temporarily store the results and models of the source domain
target_domain	Temporarily store the results and models of the target domain
train_images	The folder includes data sets of the source and target domains (Gwylab 2019)
operating_file.py	Python scripts that manipulate files, such as selecting a specified number of images from the dataset and deleting files in a specified directory
WGAN-64.py	Pre-trained WGAN
WGAN-GP-64.py	Pre-trained WGAN-GP
WGAN-64-transferlearning.py	A python script that implements a combination of transfer learning methods
WGAN-64_finetune.py	Python script to implement model fine-tuning
WGAN-64_freeze_layers.py	Python scripts to implement frozen layers