Anime Character Generation

Qi Le (le000288@umn.edu) Yuming Hu (hu000379@umn.edu)

December 18, 2020

1 Introduction

The project being conducted is based on the Generative Adversarial Networks (GANs [2]) model. The generative model, which is an unsupervised learning task that involves discovering and learning the regularities or patterns in input data, is framed as a supervised learning problem with two sub-models by GANs and the core idea of a GAN is based on the indirect training through the discriminator, which itself is also being updated dynamically. One sub-model of it is a generator model, which has been trained to generate new examples and another sub-model is a discriminator model, which has been used to classify the examples as either real or fake. We have several strategies to improve the discriminator. One technique is adversarial training, which is used to improve the robustness of the discriminator by combining adversarial attacker (generator) and discriminator in the training phase [6]. However, in this project, we will only train the two sub-models, discriminator and generator, together until the goal of the loss function are reached. WGAN-div model and StyleGAN model, which are used in the project are the modified GANs models.

The purpose of this project is to gain an understanding of the structure and theory of the advanced GANs models. We implement and optimize both WGAN and StyleGAN. We apply the noise to the WGAN, use the loss function of WGAN in StyleGAN, and further compare their performance.

2 Background and Challenges

At first, our team wanted to implement the StyleGAN2 by ourselves after learning the source code posted by NVIDIA. Then we realized that it was not applicable for two reasons. The first reason is that the project itself is very complicated and we do not have enough optimization knowledge for reproducing the project. The second reason is that the Anime Character Generation might not need the high-resolution picture to show the details of the faces. Thus, we decided to implement a simpler model first and then add some modules that we learned from the StyleGAN2 project to our model to improve the image quality.

We selected WGAN-div as our basic model from DCGAN, WGAN, WGAN-GP, LSGAN, SNGAN. Compared to DCGAN, WGAN-div has a better loss function – Wasserstein divergence, which is a smooth function. For traditional GANs (e.g. DCGAN), the generator tries to minimize the following loss function (while the discriminator tries to maximize it):

$$loss = -\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1 - D(G(z)))] \tag{1}$$

In this function, $D(x)$ is the discriminator's estimate of the probability that real data instance x is real. G(z) is the generator's output when given noise z, and D(G(z)) is the discriminator's estimate of the probability that a fake instance is real. By taking its derivative with respect to D as 0, we can get

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)} \tag{2}$$

Denote that $KL(P_1||P_2) = \mathbb{E}_{x\sim P_1} \log \frac{P_1}{P_2}$, we have

$$JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1+P_2}{2}) \tag{3}$$

Then we can obtain that

$$loss = 2JS(P_r||P_g) - 2\log 2 \tag{4}$$

When the discriminator is trained too well, the generator gradient disappears (i.e. $\nabla \to 0$), thus the generator loss cannot be reduced. When the discriminator is not well trained, the generator gradient is not accurate. We need to balance between training the discriminator too well and not well, which is in fact very difficult for GANs.

$$W(P_r, P_g) = \inf_{\gamma\sim\Pi(P_r,P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||] \tag{5}$$

Wasserstein GANs does not directly classify instances, however, it outputs the their distance (equation 5). When the P_r and P_g, especially at the beginning, do not effectively overlap, KL and JS in equation 3 is nearly 0, but the Wasserstein distance can still reflect their difference. In other words, there will a be more smooth gradient in WGANs. Of course, WGAN is also challenging, and we are still working on it.

$$\min_G \max_D \underset{G(\boldsymbol{z})\sim\mathbb{P}_g}{\mathbb{E}}[D(G(\boldsymbol{z}))] - \underset{\boldsymbol{x}\sim\mathbb{P}_r}{\mathbb{E}}[D(\boldsymbol{x})] - k \underset{\hat{\boldsymbol{x}}\sim\mathbb{P}_u}{\mathbb{E}}[\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|^p] \tag{6}$$

$$-\mathbb{E}_{G(z)\sim P_g}[D(G(z))] \tag{7}$$

$$\mathbb{E}_{G(z)\sim P_g}[D(G(z))] - \mathbb{E}_{x\sim P_r}[D(x)] - k\mathbb{E}_{\hat{x}\sim P_u}[\|\nabla_{\hat{x}} D(\hat{x})\|^p] \tag{8}$$

The WGAN-div model uses Wasserstein distance instead of the traditional distance. If we represent P_r as the real sample distribution and P_g as the sample distribution generated by the generator, $W(P_r, P_g)$, which is the result of the Wasserstein distance, can represent the minimal cost of moving Pr to Pg under the optimal path. With the Wasserstein distance, the WGAN-div model can generate a continuous loss function to evaluate the quality of the model and avoid the gradient instability that exists in the traditional GAN models [3]. Furthermore, it can get rid of problems like mode collapse, and provide meaningful learning curves useful for debugging and hyperparameter searches [1].

3 Implementation

The goal of our project is to improve the quality of the generated images. By now, there are two ways in front of us. The first way is to improve the resolution of the input and generated pictures. With the increasing resolution of the pictures, more and more details would show up on the generated pictures. The second way is the main focus of our group. We are trying to add some modules learned from the StyleGAN [5] to our basic WGAN-div models. StyleGAN has 4 main changes in the new style-based generator:

- Removing traditional input. (figure 8)
- Mapping Network. (figure 9)

- Style modules (AdaIN). (figure 10)
- Stochastic variation (Stochastic variation, generates random details for the generator by adding noise). (figure 11)

The first change is that the mapping network consisting of 8 fully connected layers is added, and the size of the output of Mapping Network W' is the same as that of the input layer (512×1). The second change is that the adaptive instance normalization is added to transform the W' vector into a style control vector and influence the generation process of the generator. The third change is that the traditional latent point input is removed. Instead of taking a point from the latent space as input, a constant 4*4*512 constant value becomes the input of the synthesis network. The fourth change is that the scaled noise is added to each resolution lever of the synthesis network.

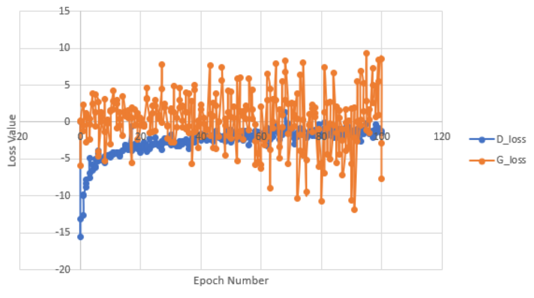
For the rest of the report, we will show the generated pictures with more details and clearer parts on the generated anime faces. Learning from the idea of StyleGAN, we add Gaussian noise to each layer of the WGAN. Further we utilize the WGAN divergence as the loss function of StyleGAN.

4 Evaluation and Discussion

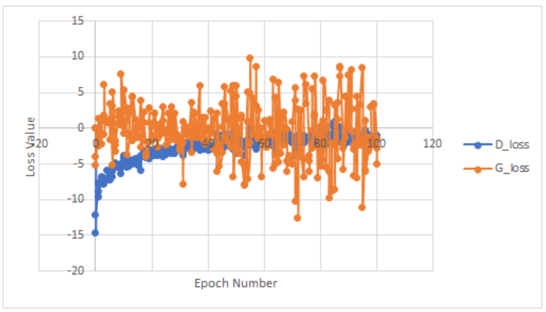
After constructing the WGAN-div model and understanding the structure of the StyleGAN model, we found that there are significant differences between these two models.

The first difference would be the structure of the generator. Our WGAN-div model contains five convolutional layers in the generator and the simpler network would connect to a more complex network until the model outputs the result. When we are trying to generate high-resolution images, this model has a big weakness, which is the long program running time. But in the generator of StyleGAN, although there are many layers, there is only one network inside the Generator. Furthermore, StyleGAN uses a smooth transition technology to handle the dynamically changing of the structure of the network during the training process.

The second difference is the normalization of the parameters in the model. The problem of difficulty in training often occurs when training GANs, because after each parameter iteration, the distribution of the output data of the previous network will change because the data have been calculated by the previous network. This phenomenon will cause difficulty for the learning of the next network. We need normalization to keep the input of each layer of the network in the same distribution during the GANs training process. In the WGAN-div model, the data would be normalized during each convolutional layer in the generator. In the StyleGAN model, the normalization almost happens in every step, such as each sampling layer, each layer in the synthesis network.



(a) WGAN wo/ loss



(b) WGAN w/ loss

Figure 1: Loss v.s. #epoch.

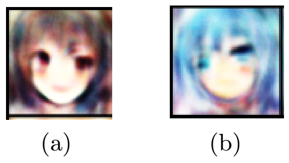


Figure 2: Images generated by WGAN without noise

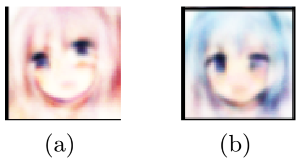
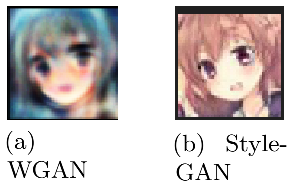


Figure 3: Images generated by WGAN with noise

The third difference is the noise. In the WGAN-div model, the noise only being generated on the input vector. But in the StyleGAN model, 10 learned per-channel scaled noises are connected to the synthesis network. Since the noise affects the quality of the images a lot, our group implements the WGAN-div model by adding the noise on every layer of the generator to see if the quality of the image increases. Both trials have been run for 100 epochs and 1115 batches with 64 batch size for each epoch. The results show in Figures 1, 2, 3. The results are similar. The main reason is that the noise added to each layer of the StyleGAN generator is scaled and transformed by learnable scaling parameters. The parameters are changed by the back-propagation and this operation would slightly change the visual expression of the features. So, a simple addition of noise does not change the quality of the generated image.

Each anime faces generated in the above two images has the resolution of 64*64 pixels. By comparing these two images in Figure 4, we can easily find out that the quality of the image generated by the StyleGAN model is much better than the quality of the image generated by the WGAN-div model in two aspects: details and decoupling. The right image has clear details, such as eyes, hair and face and the different parts of face are independent with each other, which give us the vivid image.

The official loss function for StyleGAN would be logistic loss. We modify the loss function using WGAN-div loss function. Equation 7 is used to minimize the loss function of the generator and equation 8 is used to minimize the loss function of the discriminator. Figures 7 show the running result StyleGAN using different loss functions. Both trials have been run for about 20 hours, with 32 ticks (4.48 million images). The level of detail is measured by the Perceptual Path Length. The smaller the level of detail is, the clearer the images are. We can find out that both loss functions have good performance. Compared to the level of detail of Logistic loss function, the level of detail of WGAN-div loss function is a little higher, which means for StyleGAN, WGAN-div loss function is a good choice but is not the best choice.



(a) WGAN (b) StyleGAN

Figure 4: Images generated by WGAN and StyleGAN

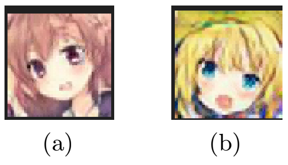


Figure 5: Images generated by StyleGAN with logistic loss function

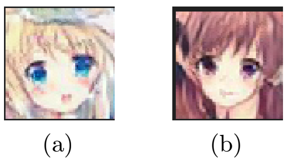


Figure 6: Images generated by StyleGAN with WGAN-div loss function

5 Conclusion

The ideas of the WGAN-div loss function, AdaIN, mapping network, upscale images, and the tricks of the convolutional kernel are very valuable. The images generated by the WGAN-div model are worse than those of the StyleGAN model because of the huge difference between the structure of the generator. A simple addition of noise to the WGAN-div model does not change the quality of the generated image. However, we obtain similar quality images from the StyleGAN model with original loss function and the StyleGAN model with WGAN-div loss function, which proves that the continuous Wasserstein distance could provide meaningful gradients. Between the WGAN-div model and the StyleGAN model, we recommend using the StyleGAN model to generate the images. With the pre-trained model, people can easily generate high-quality images from the StyleGAN model.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein gan”. In: *arXiv preprint arXiv:1701.07875* (2017).

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pages 2672–2680.

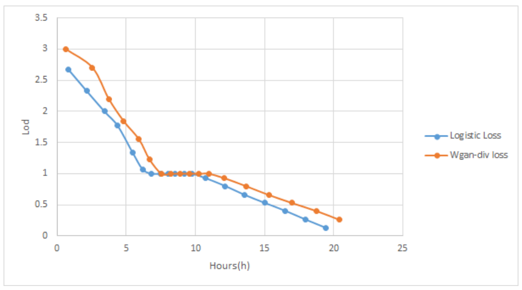


Figure 7: The Level of detail (LoD) of the StyleGAN

[3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved training of wasserstein gans". In: *Advances in neural information processing systems* 30 (2017), pages 5767–5777.

[4] Rani Horev. "Style-based GANs–Generating and tuning realistic artificial faces". In: *LyrnAI: Deep Learning Explained. Available at: www. lyrn. ai/2018/12/26/a-style-based-generator-architecture-for-generative-adversarial-networks (accessed 10 May 2019)* (2018).

[5] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pages 4401–4410.

[6] Xuanqing Liu and Cho-Jui Hsieh. "Rob-gan: Generator, discriminator, and adversarial attacker". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pages 11234–11243.

A Appended Results

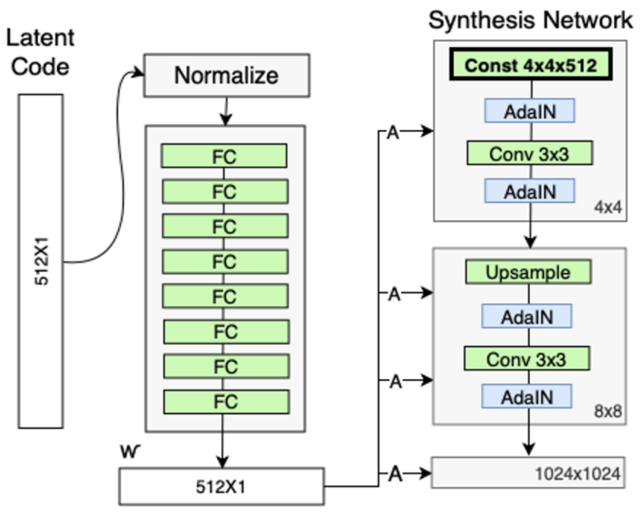


Figure 8: The Synthesis Network input is replaced with a constant input [4]

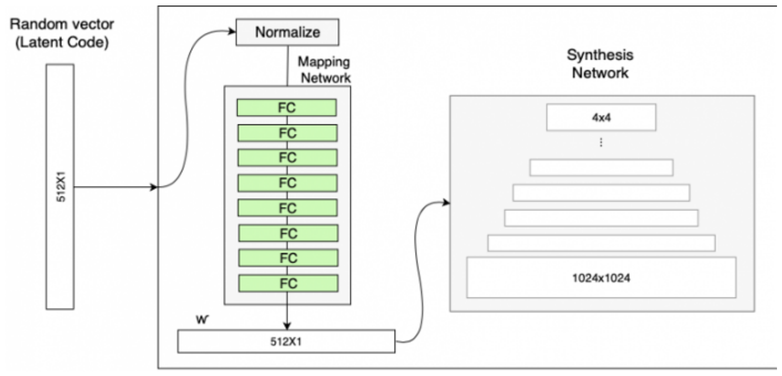
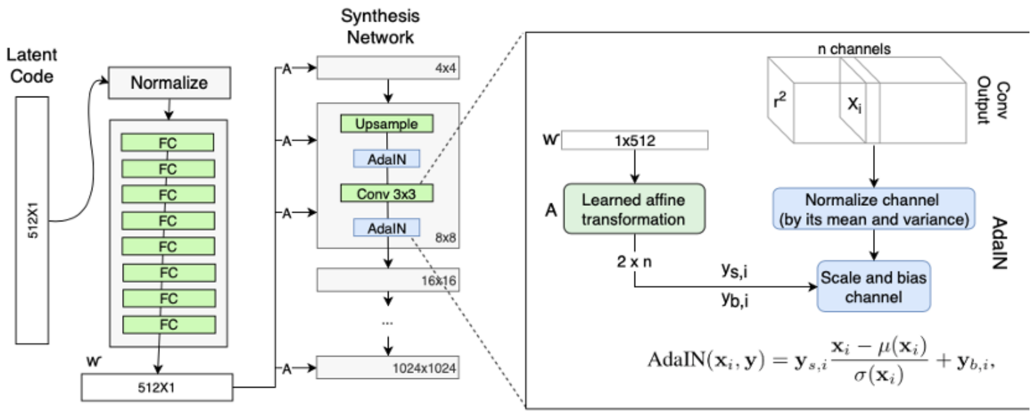


Figure 9: The generator with the Mapping Network (in addition to the ProGAN synthesis network) [4]



$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

Figure 10: The generator's Adaptive Instance Normalization (AdaIN) [4]

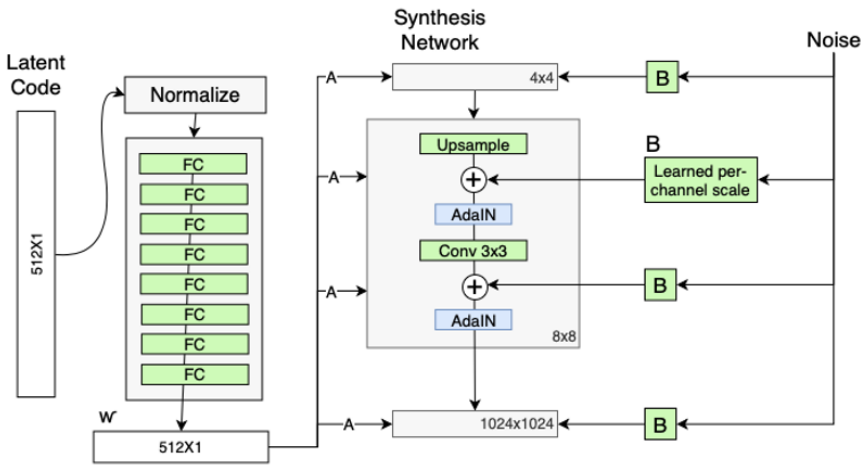


Figure 11: Adding scaled noise to each resolution level of the synthesis network [4]