Anime Character Generation

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

November 17, 2020

1 Introduction

In this project, we use Generative Adversarial Networks (GANs [1]) to generate anime pictures. We choose WGAN-div as our basic model, which leverages Wasserstein divergence as its loss function. Comparing the 64*64 resolution pictures generated from WGAN-div (figure 2) to the 128*128 resolution pictures generated from WGAN-div (figure 3), we could find a little improvement in the imaging quality. But in general, the imaging quality is not very good.

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.

3 Implementation and Discussion

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 StyleGAN2 to our basic WGAN-div models. StyleGAN2 has 4 main changes in the new style-based generator:

- Removing traditional input. (figure 4)
- Mapping Network. (figure 5)
- Style modules (AdaIN). (figure 6)
- Stochastic variation (Stochastic variation, generates random details for the generator by adding noise). (figure 7)

We want to add the first three parts to the WGAN-div model in the next steps and see how the generated pictures would change.

For the rest of the semester, we are working on improving our generated pictures to have more details and clearer parts on the faces (e.g. figure 1). Further, we are working on optimizing our work to make the generated pictures be close to the pictured generated from the StyleGAN2 model above.

References

[1] 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.

[2] 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).

A Appended Results

Figure 1: Pictures generated from StyleGAN2 model.

Figure 2: 64*64 resolution pictures generated from the WGAN-div.

Figure 3: 128*128 resolution pictures generated from the WGAN-div.

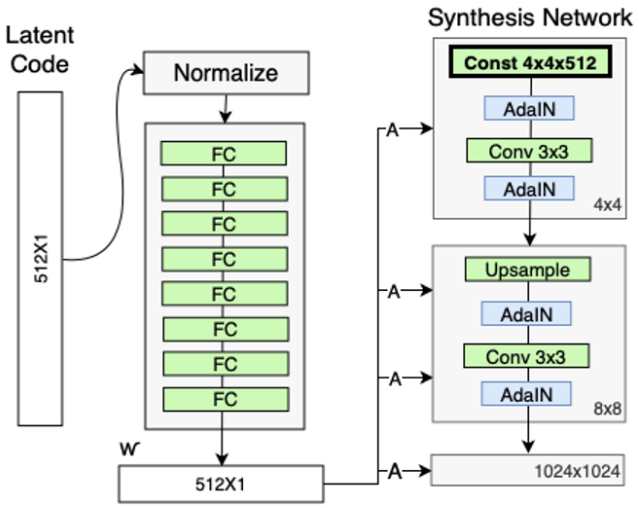

Figure 4: The Synthesis Network input is replaced with a constant input [2]

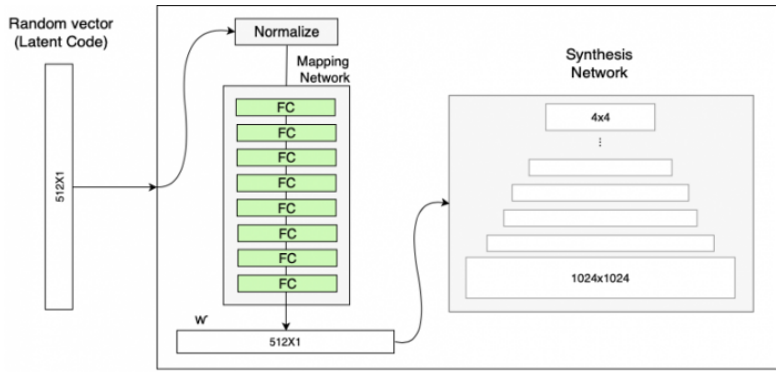

Figure 5: The generator with the Mapping Network (in addition to the ProGAN synthesis network) [2]

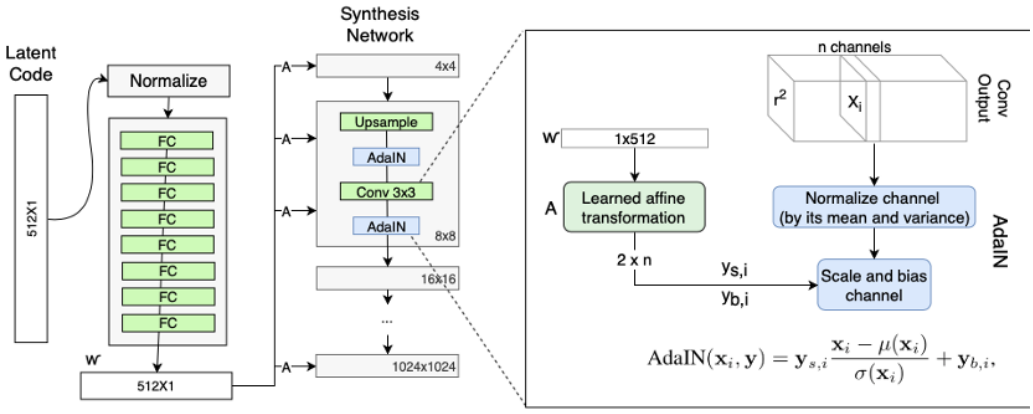

Figure 6: The generator's Adaptive Instance Normalization (AdaIN) [2]

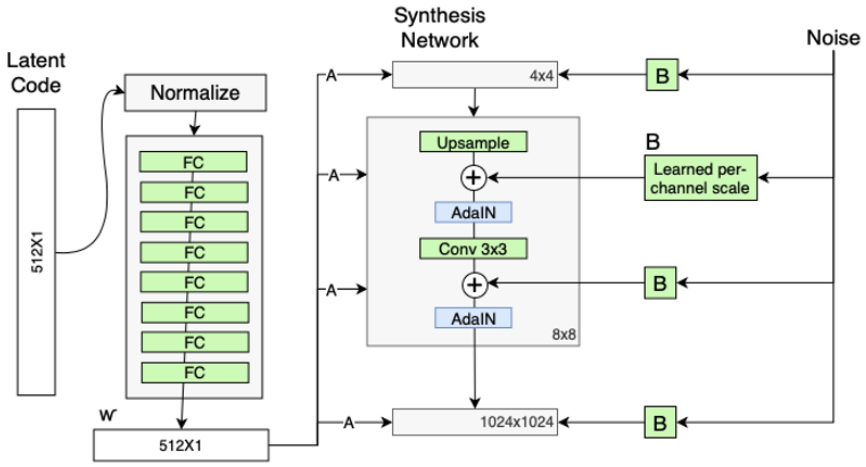

Figure 7: Adding scaled noise to each resolution level of the synthesis network [2]