Project 3

Hui Wang, s210331, Jianting Liu, s210141, Jiahao Li, s210313

1 Introduction

In this project, we use a CycleGan to convert horse to zebra and vice versa. Firstly, images of horse and zebra are extracted and resized to 128 * 128 in order to decrease training time. Then a CycleGAN network, which has two GANs and each has one Generator and one Discriminator, are implemented to update parameters and train the model. In the end, abilities of both Discriminator and Generators are strengthened that Generators can generate more real zebra images from horse(or horse images from zebra) to 'fool' Discriminator while Discriminator can distinguish better which is the real horse and which is the fake one. Different methods, such as different Gan-loss and the number of epochs are used to compare the model performance.

2 Architecture

The Generator has three parts: one encoder, one transformer and one decoder. First, we input resized images(128*128) to the Encoder. The encoder is composed of four layers, which extract images' features through convolution. Then, The result will be passed to Transformer, which has 6 ResNet Block. Each ResNet Block has 2 convolutional layer, which can retain the original image features during conversion. At last, restore features from transformation's result by Decoder and use one output layer to produce the final RGB image.

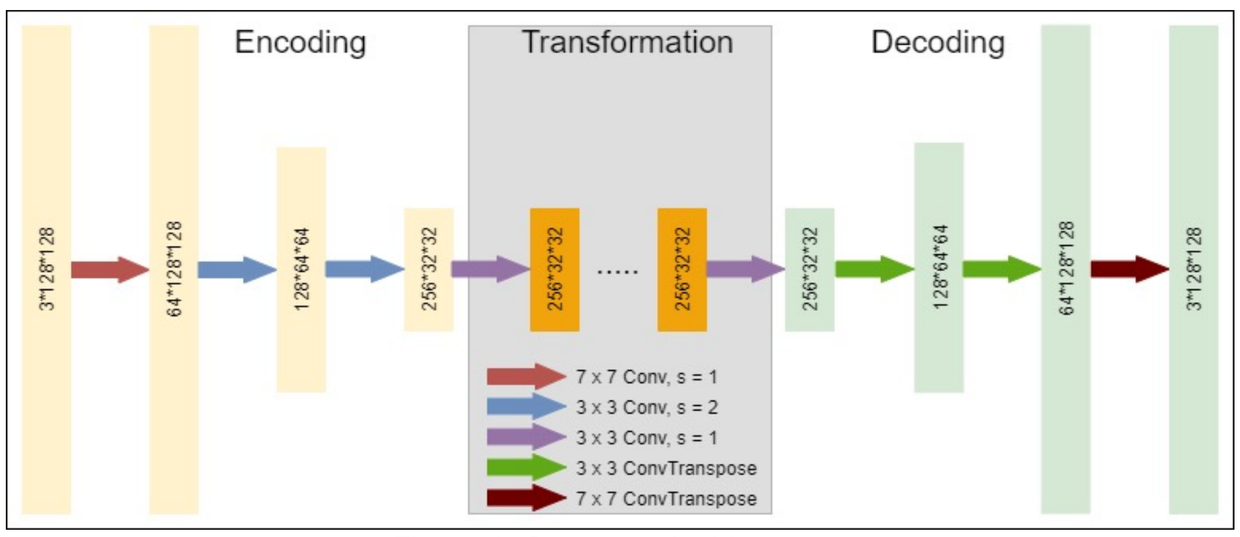

Figure 1. Generator Architecture

The Discriminator has a fully convolutional network. First, we use 4*4 kernel to convolution the image and the first 3 times stride are 2, the following two times' stride are 1, which decrease the size of the image and increase the channels. It would be more efficient than computing the whole image and make discriminator to focus on surface-level features.

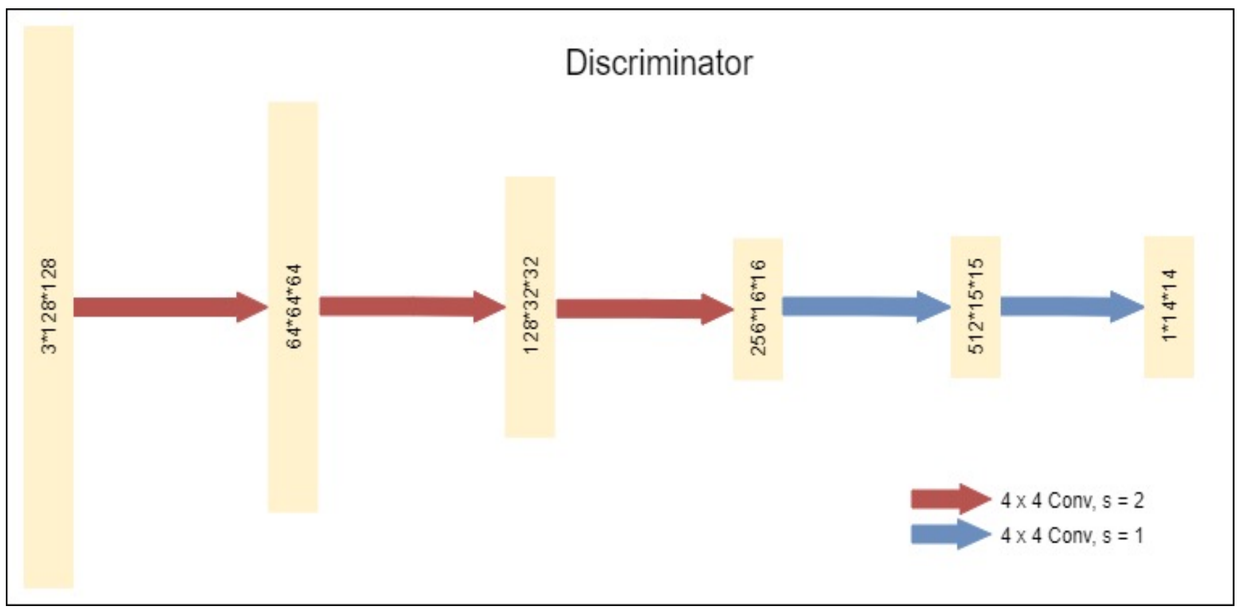

Figure 2. Discriminator Architecture

3 Performance

We use following measures to see the different performance: different GAN-Loss, different number of epochs

3.1 Performance with different GAN-Loss

There are two types of CycleGAN we used here. LSGAN uses a least square loss. We also choose MINIMAX-GAN to check its performance.

LSGAN Loss:

$$minV(D) = \frac{1}{2}E_{x\sim p_{data}(x)}[(D(x)-b)^2] + \frac{1}{2}E_{z\sim p_z(z)}\left[\left(D\left(G(z)\right)-a\right)^2\right]$$

$$minV(G) = \frac{1}{2}E_{z\sim p_z(z)}[(D(G(z)-c)^2]$$

Here we choose a=0, b=1, c=1.

MINIMAX-GAN Loss:

$$maxV(D) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}\left(log\left(1-D\left(G(z)\right)\right)\right)$$

$$minV(G) = E_{z\sim p_z(z)}\left(log\left(1-D\left(G(z)\right)\right)\right)$$

For Generator, we also need to apply Cyclic-Consistency Loss and Identity Loss.

Cycle-Consistency Loss:

$$E_{x\sim p_{data}(x)}\left\|F\left(G(x)\right)-x\right\|_1 + E_{y\sim p_y(y)}\left\|G\left(F(x)\right)-y\right\|_1$$

Identity Loss :

$$E_{x\sim p_{data}(x)}\|F(x)-x\|_1 + E_{y\sim p_y(y)}\|G(y)-y\|_1$$

The result of model using LSGAN is shown in figure 3.1 and result of model using MINIMAX GAN is shown in figure 3.2

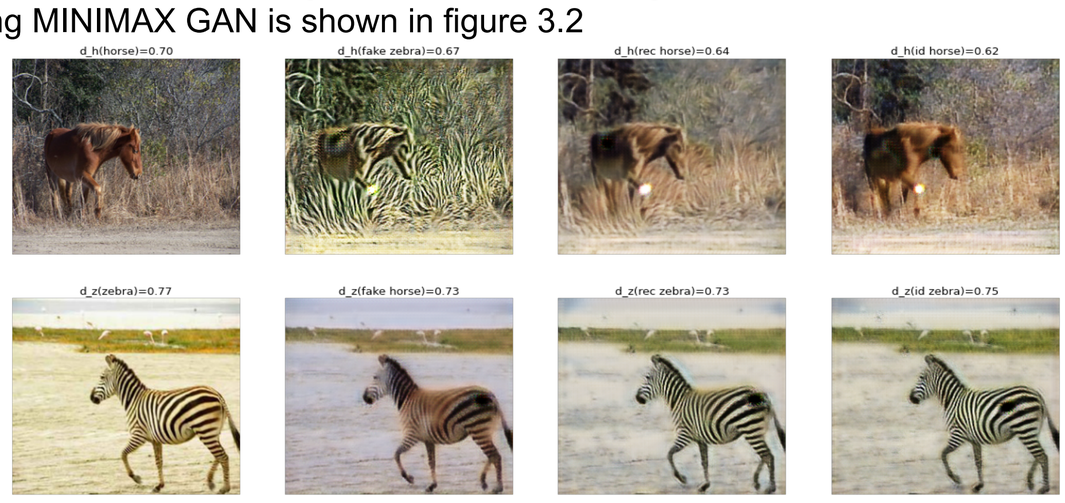

Figure 3.1 Performance of LSGAN with 10 Epochs

Figure 3.2 Performance of MINIMAX GAN with 10 epochs

3.2 Performance with different number of epochs

We set different number of epochs with the same LSGAN, range from 5 to 20, to see the performance and the results are shown below.

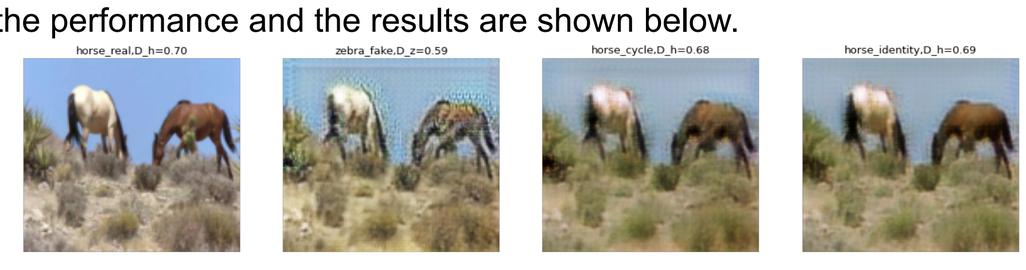

Figure 4.1 Performance after 5 epochs

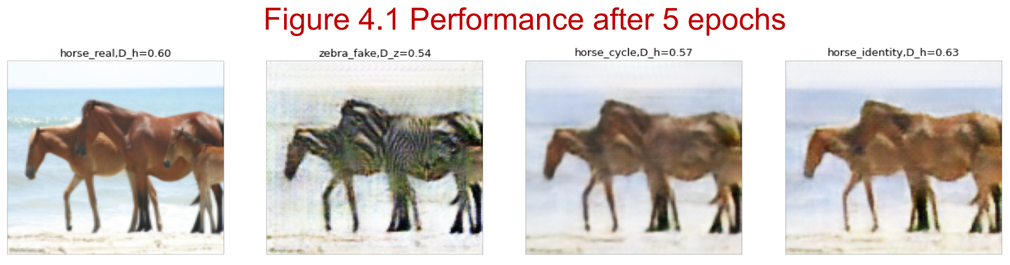

Figure 4.2. Performance after 20 epochs

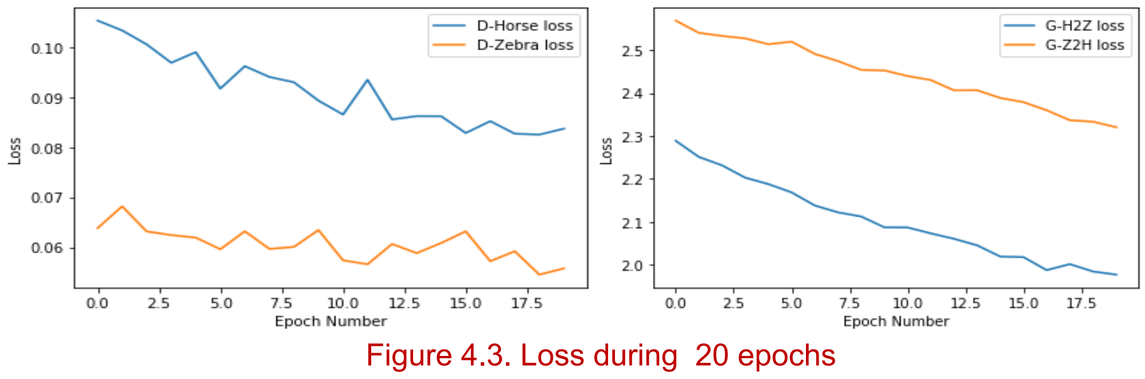

Figure 4.3. Loss during 20 epochs

3.3 Frechlet Inception Distance

Here we use FID to evaluate the quality of produced image.

FID between real_horse and fake_horse	FID betweem real_zebra and fake_zebra
68.94957637346	72.8889237902963