Use CycleGAN to Generate Monet-style Painting
Intro. to Artificial Intelligence Final Project

Hong-Pei Chen 109550006 50%

Yi-En Li 109550090 50%

Abstract

Our project is based on one competition on Kaggle: "I'm Something of a Painter Myself – Use GANs to create art - will you be the next Monet?". The topic has raised our great interest, and therefore decided to dive into this topic, trying to build GAN to generate Monet-style photos.

1. Introduction

Computer vision has advanced tremendously in recent years, and GANs are now capable of mimicking objects in a very convincing way. However, by using GAN only, the result may have distorted from the original one, since there is less limitation for the generation. Therefore, we have decided to use Cycle-GAN for our implementation. For one thing, by including cycle consistency, the generated results can be more related to input, since they can be easier to reconstruct. For another thing, Cycle-GAN can achieve two-way transformation, which provides more convenience and flexibility when translating images.

2. Related Work

Generative Adversarial Networks (GANs) have achieved impressive results in image generation, image editing, and representation learning. However, CycleGAN, which adopted the idea of adversarial loss in GAN, brought in a new idea of cycle consistency loss and has made an improvement in the field of image generation, referred to the paper [2]. When trying to implement the algorithm, we have referred to the tutorial on kaggle [1] and made several revisions, including the structure of the network architecture, the dataset source, and the loss function.

3. Methodology

First, we prepare our dataset by using Tensorflow's built-in datasets (cycle_gan/monet2photo). The original size is 286*286. we make the input size become 256*256 by randomly flipping and cropping for the purpose of data augmentation.

Second, we construct the building blocks used in the CycleGAN generators and discriminators, including the padding method(ReflectionPadding2D), residual blocks, and the downsample and upsample implementation. Then we construct the generator with two downsampling blocks, nine residual blocks, two upsampling blocks, and construct the discriminator following the architecture: $C64 \rightarrow C128 \rightarrow C256 \rightarrow C512$. Finally, we define gen_G, gen_F, disc_X and disc_Y, which would be the key elements in CycleGAN.

Third, we build CycleGAN for our training process. We construct the generators and discriminator for both monet and photo. By maintaining the consistency, we would use a generator G to translate a monet to photo, and another generator F to translate it back to the monet to make sure the image will be as similar as possible. Discriminator will need to distinguish both images translated by generators. The loss function is below(Figure 1.). We use mean square error to calculate the loss for Loss-GAN and mean absolute error to calculate the loss for Loss-cycle and Loss-identity. In particular, for Loss-GAN, we need to train G to minimize the $E_{x \sim P_{data}(x)}[(D(G(x)) - 1)^2]$ and train the D to minimize $E_{y \sim P_{data}(y)}[(D(y)1)^2] + E_{x \sim P_{data}(x)}[D(G(x))^2]$. [2]

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

$$\mathcal{L}_{\text{identity}}\ (G, F) = \mathbb{E}_{y \sim p_{\text{data}}\ (y)}\ [\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}\ (x)}\ [\|F(x) - x\|_1]$$

$$Loss = Loss_{GAN} + Loss_{cycle} + Loss_{identity}$$

Figure 1. Loss function

Finally, we train our model with 300 epochs, which took about 11 hours. For each epoch, the callback function would save the checkpoints file and randomly choose 4 images from the dataset to monitor the training process.

Figure 2. CycleGAN result for different epochs

4. Experiments

1. **Different epochs**

   We use 50, 100, 200, and 300 epochs to see the difference in the translated monet-style photo.

   With different epochs, we could see these photos becoming more and more similar to oil paintings with the obvious brush strokes ,and it seems like there are some differences.

   - Tone

     In 50 epochs, it mixes more purple tone. However, in 100 and 200 epochs, the tone becomes green. In the 300 epochs, the tone is more similar to the original photo but mixed a little red.

   - Dark area

     In left row 1 and right row 3, the color scale of the sky changes significantly. It may be possible that the model is sensitive to the huge cover of the dark area.

2. **GAN vs CycleGAN**

   The images translated by GAN also have the obvious brush strokes and the feeling of monet-style. However, it has some distortions. Some of the objects are totally different from the original ones. In row 1, the mountain disappeared. In row 5, the houses were translated to forest by GAN. The image's color is also paler than that translated by CycleGAN. Therefore, the CycleGAN is a key point to maintain the accuracy of the transformation.

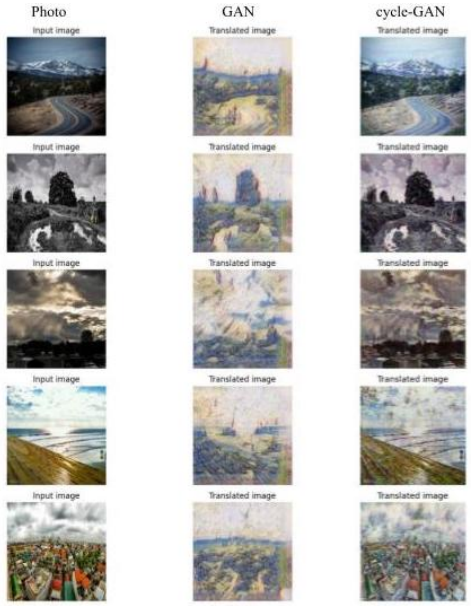

Figure 3. GAN vs CycleGAN

5. Information

Github link: https://github.com/HopePei/NYCY-2022-AI-Final-Project

References

[1] Amy Jang. Monet CycleGAN Tutorial- I'm Something of a Painter Myself. 1

[2] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. 1