Individual Final Report

Introduction

The project aims to help black and white pictures to get colors. This technology could be further applied in real life, like restoring old images. Moreover, we could also apply it to the 2D animation production industry that helps complete the work which needs the amount of mutual coloring in a short time. Shared work is the final presentation, final report, literature review and data preprocessing, determining which model to use.

Individual work

My work is to develop a structure of generative adversarial networks to colorize the grayscale photos. After the group discussion, I do some research on color space and GAN. I found many different color space and I decided to use L*a*b* color space because it is a color space that is close to what our eyes see. The L represent lightness and the a* is the combination of red & green value, the b* is the combination of blue & yellow value. The generator is an encoder-decoder structure and the discriminator is a CNN structure that was used in Deep Convolutional GAN.

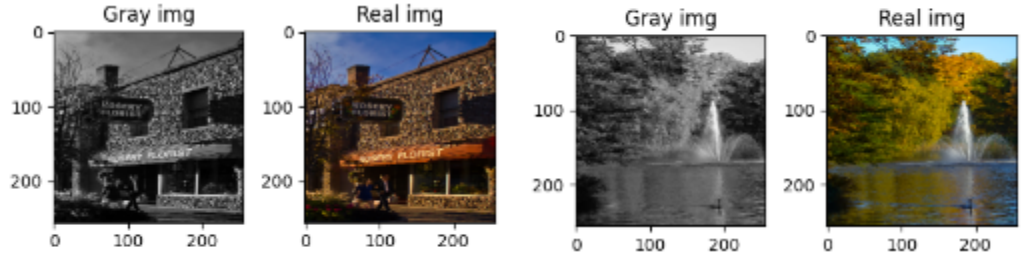
Portion of the work

I write the GAN based on L*a*b* color space. For data preprocessing, I use cv2 to read the image data and then resize to (256, 256). Afterwards, I transform them to the L*a*b format data. And pick only L* channel data as a single dataset for the input of the generator. And the a*b* channel data as the real label for discriminator. The structure of the generator is inspired from a paper (https://arxiv.org/abs/1803.05400). In their baseline work, they use an encoder-decoder structure on CNN to colorize images. So I use a similar structure on the generator. The discriminator is a CNN that was used in Deep Convolutional GAN. The training processing was very hard. As the common problem with GAN, the training process is very slow and unstable. In the first several times, the loss of discriminator would decrease very quickly. So I use a less powerful discriminator to try to make the generator perform better in the game. And it did. Then I add more state parts which is shown below.

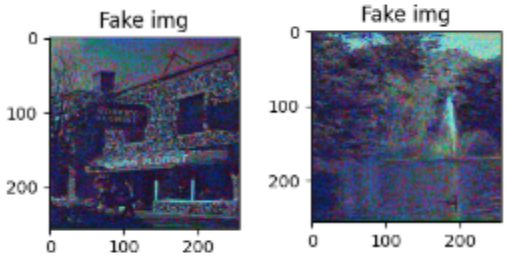
```
nn.Sequential( # Sequential,
    LambdaMap(lambda x: x, # ConcatTable,
        nn.Sequential( # Sequential,
            nn.Conv2d(128,128,(3, 3)),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.Conv2d(128,128,(3, 3)),
            nn.BatchNorm2d(128),
            ),
        shave_block(2),
        ),
    LambdaReduce(lambda x,y: x+y), # CAddTable,
    ),
```

For this part, the input and output number is the same. This block allow generator learn deeper inside the image data (transformation from L* to a*b*). After adding 5 state blocks, the generator begins to generate good images which are similar to the real image. Then I train more epochs, the same

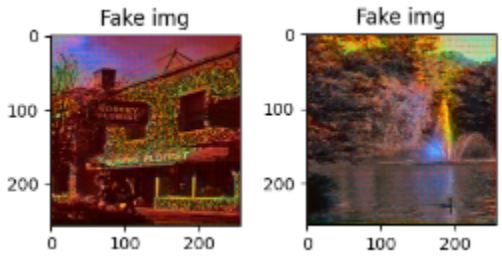
Results



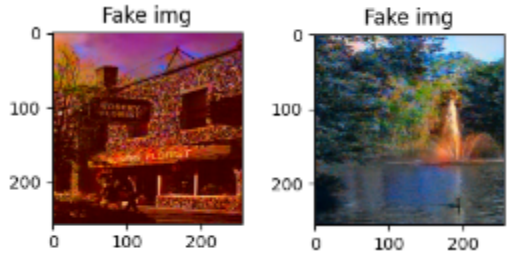
The result of 1 epochs training is shown below. Model learn a little bit of a*b* channel but there is still a lot of noisy data inside the image.



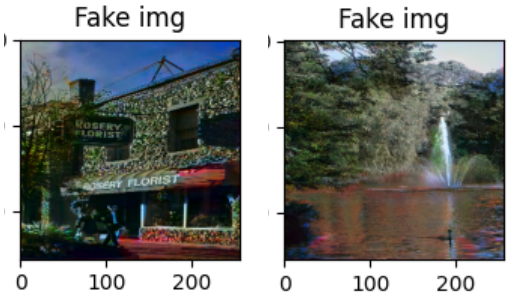
After 2 epochs training, the result is much better than the first epoch. There is still some noisy data in the image but not that much like the first epoch.



When it comes to 50 epochs, there is no more noisy data in images. But the color is still not as good as real images.



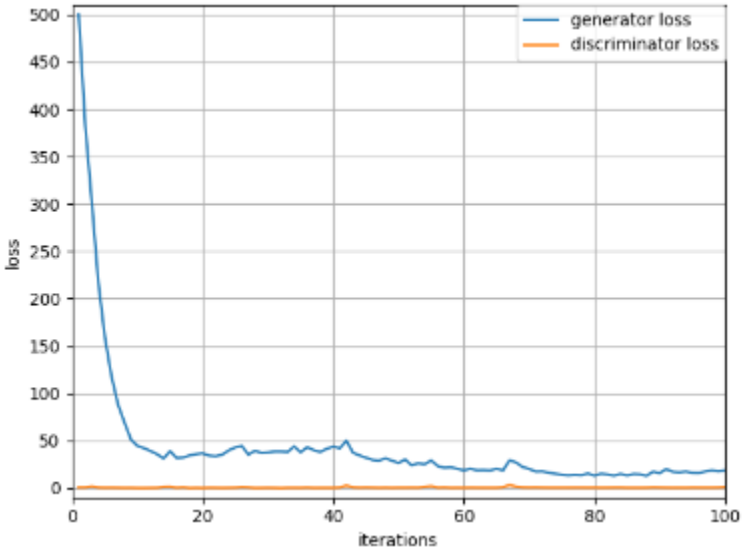
Here is the result for 100 epochs training, the color is very similar to the real image.



After 200 epochs training, the color of images are much better than previous. And even better than the original image.

The result of 300 epochs training is shown below, images are very close to the original one.

The loss function of L*a*b* GAN is shown below. The training processing is stable. When training after 10 epochs, the loss goes down to below 50 and then keeps decreasing.



Summary and conclusions

I develop a GAN to colorize grayscale images. The result is quite good on scenery photos. But not that good on photos with people. This is because our dataset contains more scenery photos than people's. I learnt a lot about pytorch framework, network design, especially for generative adversarial networks. And I learnt a lot about how to train GAN stably like flip labels, soft & noisy labels. Note that the model is not well-trained due to the computational power. For the improvement, I will use a larger dataset with balanced categories. I would seek a better quantitative metric to measure performance such as peak signal-to-noise ratio (PSNR) and root mean square error (RMSE)

Percentage of my own code: 1-224/757=70.41%

References

Image colorization: https://arxiv.org/abs/1803.05400

Deep Convolutional GAN: https://arxiv.org/abs/1511.06434