Deep Learning Practice Lab 6

0856056 Yu-Wun Tseng

May 2020

1 Introduction

The goal of this lab is to implement a conditional Generative Adversarial Networks(GAN) for image generation and classification. The training input are images with corresponding object labels, each image has at most three objects and at least one object. The object has eight colors and three shapes. Given the object condition and the image, the discriminator should classify which objects exist in the image. And given the object condition and noise, the generator should generate the synthetic images with corresponding objects.

2 Experiment Setups

This lab is done by using Pytorch. The procedure follows the pytorch official DCGAN tutorial [1].

2.1 Model Details

The architecture of conditional GAN is simply concatenate the condition with images and the latent variable. The condition is embedded to a one hot vector. For the discriminator, the condition is passed into a linear layer, the output size is the square of the image size. After then, the condition is viewed as an (image size, image size) array, e.g., (64, 64) in my model. Finally, the condition is concatenated after the image, as a result, the image has four channel. For generator, the condition is concatenated to the latent variable directly.

The GAN architecture is DCGAN [2], the generator is composed by deconvolution layers, batch normalize layers and ReLU layers, the architecture is shown on Figure 1. And the discriminator is composed by convolution layers, batch normalize layers and LeakyReLU layers.

In order to fetch the data, I implement a dataloader. Each training data is a pair with image and object label. The images are resized and crop to image size which is 64. After convert the image to tensor, I normalize the tensor with mean and variance equal to 0.5. The object labels are converted into one hot vectors. In addition, I shuffle the data when training.

The loss function is BCE, the optimizer is Adam. Additionally, I use learning schedule to decay the learning rate. The scheduler is StepLR.

2.2 Hyperparameters

The latent size is 100, the batch size is 128, and the epochs is 700. The learning rate of generator is set to 2e-4 and the learning rate of discriminator is set to 1e-4 in the beginning. Every 50 epochs, both learning rate of generator and discriminator will be multiplied by 0.5. In addition, the image size is 64, and both feature map depths of generator and discriminator are 64.

3 Results

The generated testing image are shown on Figure 2. As the figure shows, the generator can generate most images with correct objects corresponding to the condition, but the objects are a little blurry. However, if more objects in the condition that need to be generated, the model will not work well but will generate more noise, such as no clear-shaped color block. Additionally, the highest score is 0.62.

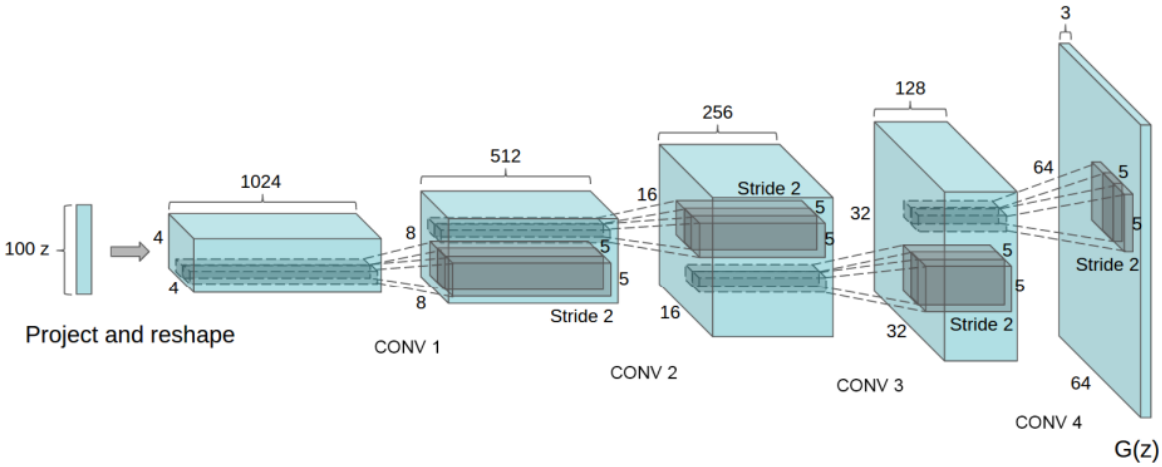

Figure 1: Generator Architecture

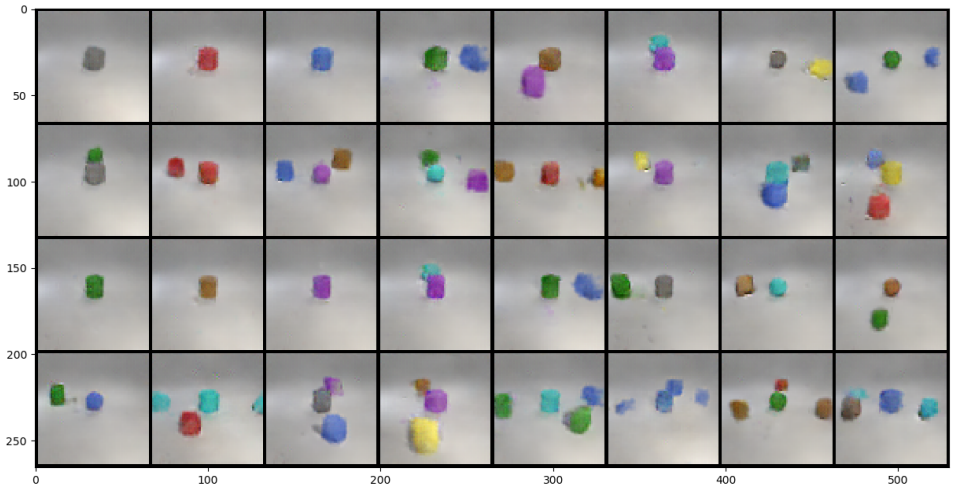

Figure 2: Generated Images

4 Discussion

The GAN training time is very long. It took a lot of time to verify whether the model works well or whether the methods are useful. First, I used the exact same architecture as the tutorial to train the model. After many hours, I found the loss of discriminator was very low and the loss of generator was much higher than it. As a result, I set the discriminator learning rate to half of the generator learning rate. The model perform much better, however, the accuracy difference of each epoch varies greatly. And after I use learning rate schedule, the accuracy trend is more stable.

References

[1] Nathan Inkawhich. Dcgan tutorial.

[2] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, November 2015.