conditional Generative Adversarial Networks (cGAN)

David Ishak Kosasih
20195033

For cGAN implementation in python, I used example from the following website : https://github.com/znxlwm/tensorflow-MNIST-cGAN-cDCGAN/blob/master/tensorflow_MNIST_cGAN.py.

When implementing this code, I added several lines for saving the model. This cGAN implementation, generated images and the trained model can be found on my GitHub : https://github.com/ishakdavidk/Machine-Learning/tree/master/Homework%203.

The trained models were saved by using the following functions :

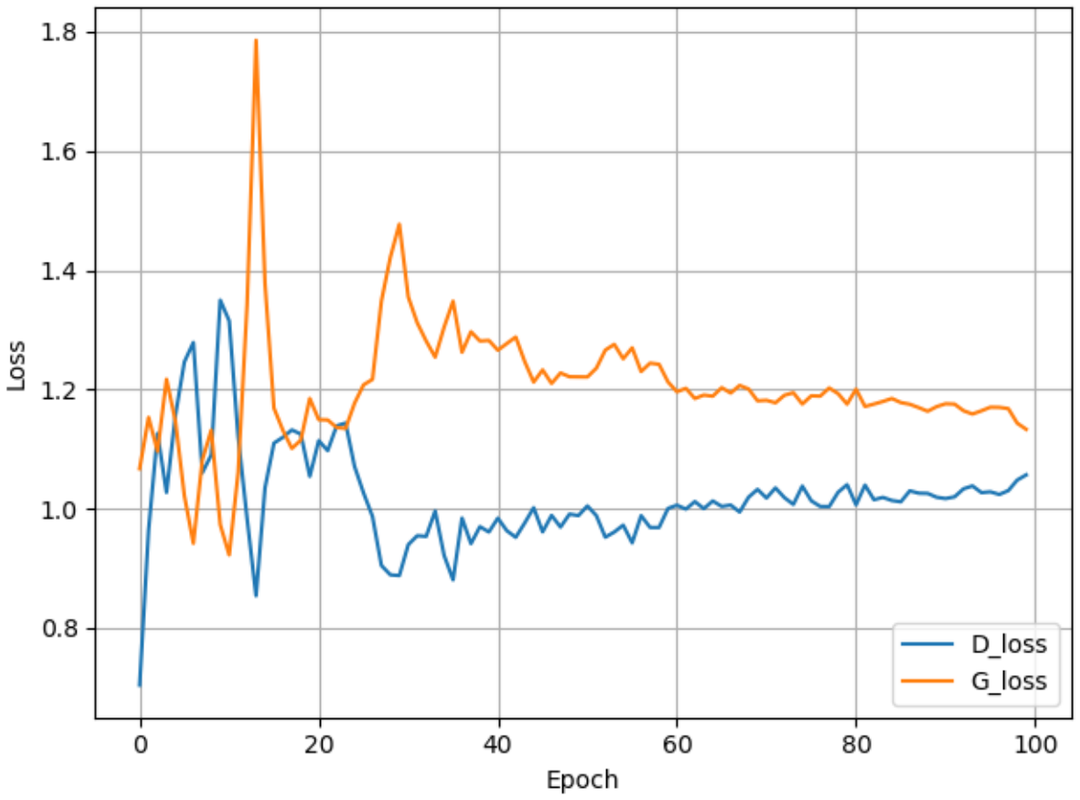
```
saver = tf.train.Saver()
saver.save(sess, 'MNIST_cGAN_model')
```

This model can be restored by using the following example commands :

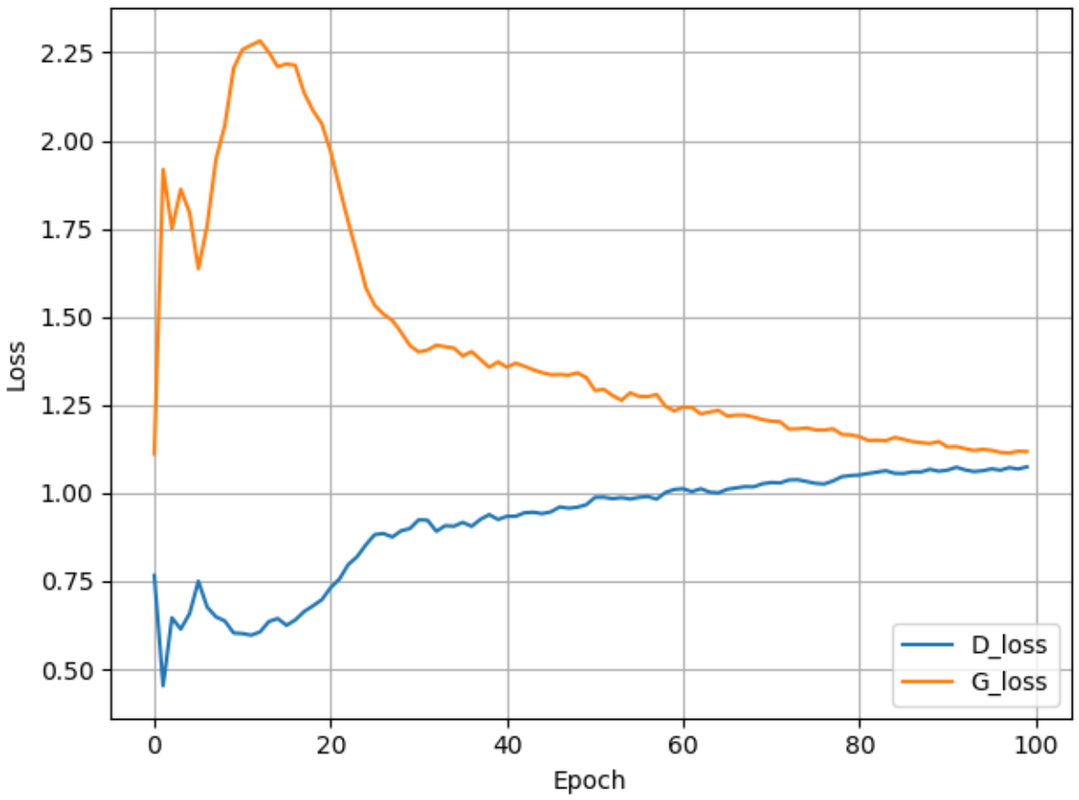
```
import tensorflow as tf

sess=tf.Session()
saver = tf.train.import_meta_graph('MNIST_cGAN_model.meta')
saver.restore(sess,tf.train.latest_checkpoint('./'))
```

Loss Function Plot for MNIST

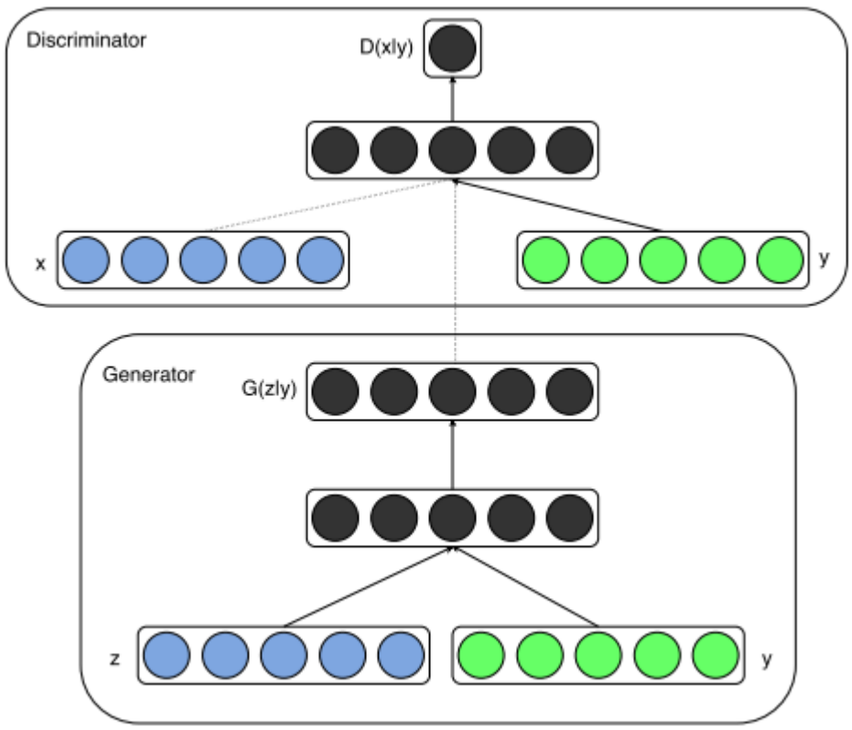


Loss Function Plot for Fashion MNIST



Analysis

GAN and cGAN basically have the same structure. The only different is that in cGAN (as depicted in the figure bellow), we add some extra information y (can be class labels) to the input of both generator and discriminator.



The objective function of cGAN is also very similar to the basic GAN.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

In this loss function generator tries to minimize log(1 − D(G(z|y)) while discriminator tries to minimize logD(x|y).

After running this code, it can be seen that the generator could produced reasonable result after around 25 epochs and good result after approximately 80 epochs for both MNIST and fashion MNIST dataset. If we observe our plots, it is clear that after 80 epochs, both generatoer and discriminator approached similar values, meaning it managed to improve each other.