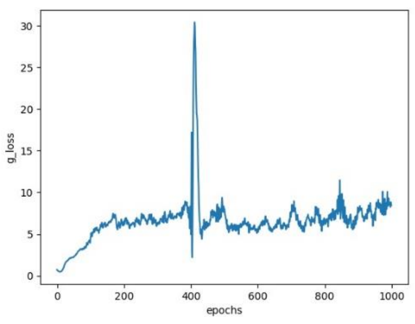
Project report 2-Yubo Wang (5334-5862)

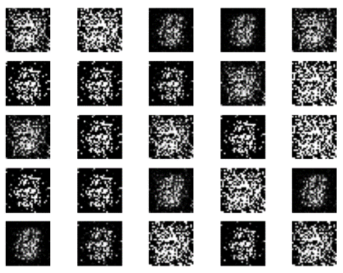
- **GAN: Adam optimizer (learning rate 0.0005), binary cross-entropy, epochs=1000, batch_size=200, 10000 images**

For the generator, I use four full-connection layers, the first three contain 64, 128, 256 nodes respectively, and the last one contains 28*28*1=784 nodes, which is the size of the output image. I also use three activation layers, all with Leaky ReLU function, and three batch normalization layers. The connectivity is shown as follow:
dense(64)->LeakyReLU->batch_normalization->dense(128)->LeakyReLU->batch_normalization->dense (256)->LeakyReLU->batch_normalization->dense(784)->reshape(28*28*1)

For the discriminator, I use three full-connection layers, contain 128, 64, 1 node(s) respectively, two activation layers all with Leaky ReLU function, and one flatten layer at the first of the model. The connectivity is shown as follow:
flatten->dense(128)-> LeakyReLU->dense(64)-> LeakyReLU->dense(1)

The result of the GAN are shown as follow: (average generator loss around 6.548)



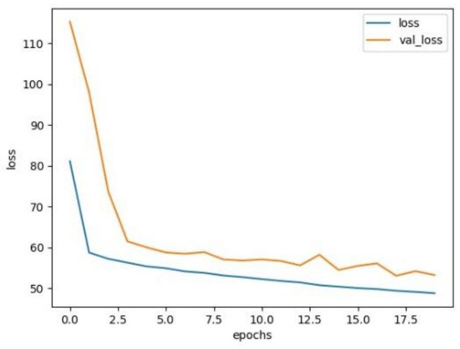


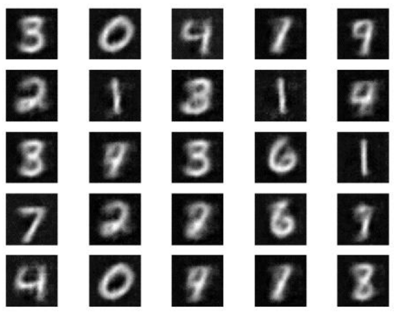
- **VAE: Adam optimizer (learning rate 0.0005), epochs=20, batch_size=64, validation_split=0.2, 10000 images**

Because the GAN result is not decent, I decided to try some more complicated model structure in the VAE model. For the encoder, I use five convolution 2D layers with 1, 32, 64, 64, 64 filters (the dimensionality of the output space), respectively. I also use five activation layers, all with Leaky ReLU function, and five batch normalization layers. At the last of the model, I use one flatten layer, one lambda layer, to add some noises to the input and two dense layers that calculate the mean and the log variance. The connectivity of the model is shown as follow:
Input(28*28*1)->Conv2D(1)->batch_normalization->LeakyReLU->Conv2D(32)->batch_normalization-> LeakyReLU->Conv2D(64)->batch_normalization->LeakyReLU->Conv2D(64)->batch_normalization->LeakyReLU->Conv2D(64)->batch_normalization->LeakyReLU->flatten->dense(2)(mean)/dense(2)(variance) ->lambda_output(2)

For the decoder, I use four transposed convolution 2D layers with 64, 64, 64, 1 filter(s) respectively to decode, three batch normalization layers, and four activation layers all with Leaky ReLU function. The connectivity of the model is shown as follow:
Input(2)->dense(64)->reshape(shape_before_flatten)->Conv2DT(64)->batch_normalization>LeakyReLU->Conv2DT(64)->batch_normalization->LeakyReLU->Conv2DT(64)->batch_normalization->LeakyReLU->Conv2DT(1)->batch_normalization->LeakyReLU

The result of the VAE are shown as follow: (average loss: 54.052, average validation loss: 62.625)





Because the generated numbers already seem pretty good, I plan to increase the training dataset and try different training parameters to increase the accuracy.