COP 6610 Machine Learning Fall 2020
Project Report

Nikunj Sarda
UFID: 93606581
nikunjsarda@ufl.edu
December 12, 2020

1 Introduction

The objective of the project is to setup GAN and VAE on MNIST handwritten datasets to generate "fake" handwritten digit.
The original idea of GAN was proposed by Goodfellow et al in 2014 [1], and grow rapidly into lots of different variations, including DCGAN(Deep Convolutional GAN) [2], LSGAN(Least Square GAN) [4], Softmax GAN [6] and so on.
The GAN consists of discriminator to distinguish between fake and real digits and generator to generate digits using the inputs. After training the discriminator and generator for several iterations, if converge the generator will produce same outputs as the inputs and discriminator will not be able to distinguish between generator output and the input.
Variational Auto Encoder(VAE) was proposed by Kingma and Welling in 2013 [5]. It will have decoder, an encoder and a function to measure loss. Both decoder and encoder are two separate neural networks, the purpose of encoder is to convert input image to latent space and the purpose of decoder is to output image after reconstructing it form latent space. The training of VAE is done over the parameters of distribution on representation i.e. it allows error or noise, which helps it to avoid overfitting.

2 Model Descriptions

2.1 MNIST DATA

MNIST datasets consists of handwritten digits between 0 and 9 in the form of 28X28 pixel grayscale images total of 70K.
Using the keras to get the MNIST dataset, returning two sets of data consisting of input and output for training data and another set of input and output for test data.

2.2 GAN

2.2.1 Discriminator Model

The model architecture consists of total 8 layers, namely, 2 convolution layer with 64 filters each, two layers of LeakyReLu with alpha as 0.2, two layers of Dropout, one layer of flatten and one layer of Dense with sigmoid function activation. Using the gradient descent of version Adam with learning rate 0.0002 and momentum of 0.5.
So If we plot the model will look like below image:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 14, 14, 64)        640
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 7, 7, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 3137
=================================================================
Total params: 40,705
Trainable params: 40,705
Non-trainable params: 0
_________________________________________________________________
```

Figure 1 Layers of Discriminator Model GAN

2.2.2 Generator Model

The model architecture consists of 8 layers, namely, dense, leakyReLu, reshape, convolution transpose and convolution. So if we plot the model it will look like below image:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 6272)              633472
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 6272)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 14, 14, 128)       262272
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 128)       262272
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 1)         6273
=================================================================
Total params: 1,164,289
Trainable params: 1,164,289
Non-trainable params: 0
_________________________________________________________________
```

Figure 2 Layers of Generator Model GAN

2.2.3 Training

The discriminator is trained with both real and fake samples, real is generated by randomly sampling the training datasets, while fake is generated by randomly generating random pixels' values.
The training of the generator depends on the discriminators performance. If the discriminator is good the generator is updated more often as compared to other scenario.

2.2.4 Results

Adding part of the results for the GAN.

Figure 3 Epoch = 10

Figure 4 Epoch = 100

Figure 5 Epoch = 200

Figure 6 Epoch 300

Figure 7 Epoch 400

Figure 8 Epoch 500

2.3 VAE

2.3.1 Encoder

The model architecture consists of 8 layers, namely, input, convolution, flatten, dense and sampling. On plotting the model, it will look like the below image:

```
Model: "encoder"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 28, 28, 1)]  0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 14, 14, 32)   320         input_1[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 7, 7, 64)     18496       conv2d[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 3136)         0           conv2d_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 16)           50192       flatten[0][0]
__________________________________________________________________________________________________
z_mean (Dense)                  (None, 2)            34          dense[0][0]
__________________________________________________________________________________________________
z_log_var (Dense)               (None, 2)            34          dense[0][0]
__________________________________________________________________________________________________
sampling (Sampling)             (None, 2)            0           z_mean[0][0]
                                                                 z_log_var[0][0]
==================================================================================================
Total params: 69,076
Trainable params: 69,076
Non-trainable params: 0
```

Figure 9 Layers of Encoder Model

2.3.2 Decoder

The model architecture consists of 6 layers as shown below:

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense_1 (Dense)              (None, 3136)              9408
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        36928
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 32)        18464
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 1)         289
=================================================================
Total params: 65,089
Trainable params: 65,089
Non-trainable params: 0
```

Figure 10 Layers of Decoder Model

2.3.3 Training

The training for VAE is done by a custom method with batch size as 128, using gradient descent version of Adam.

2.3.4 Results

Adding the part of the result plotted in grid format, showing the training result after different epochs.

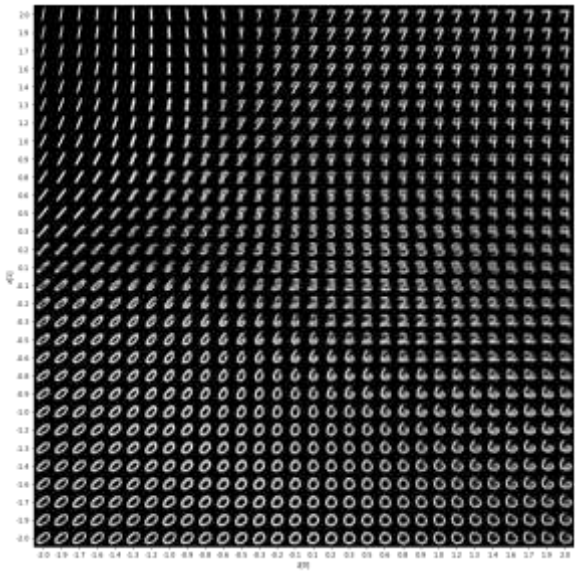

Figure 11 Epoch = 10

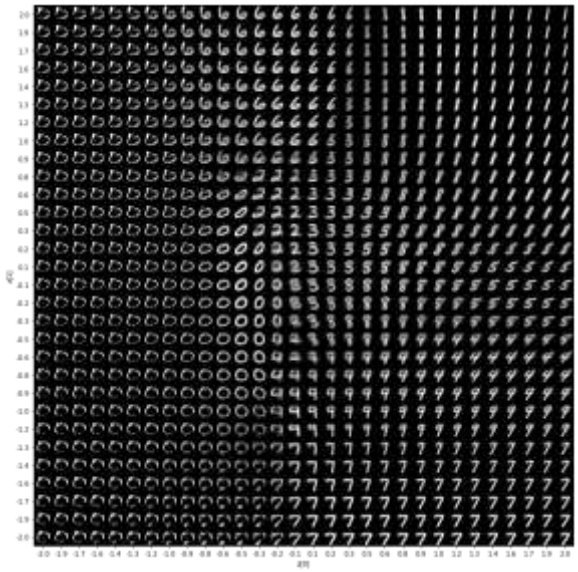

Figure 12 Epoch = 100

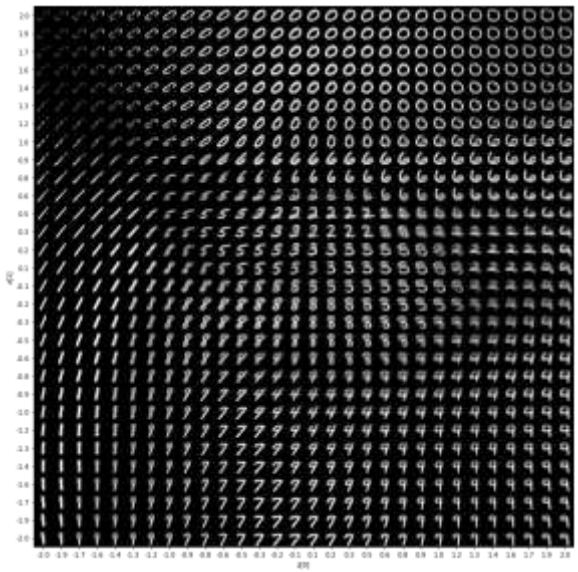

Figure 13 Epoch = 200

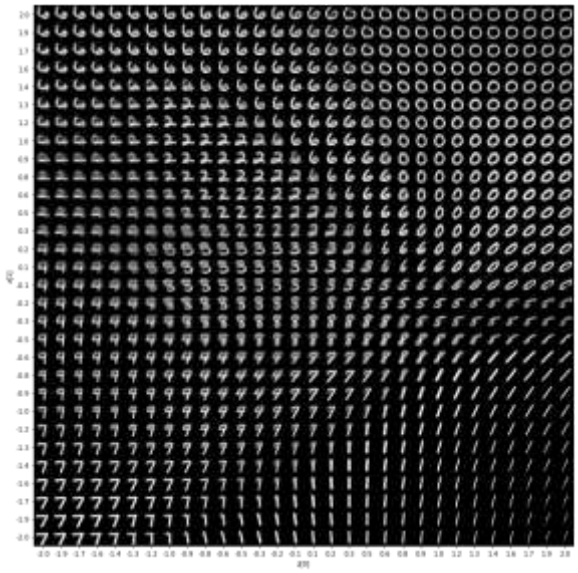

Figure 14 Epoch = 300

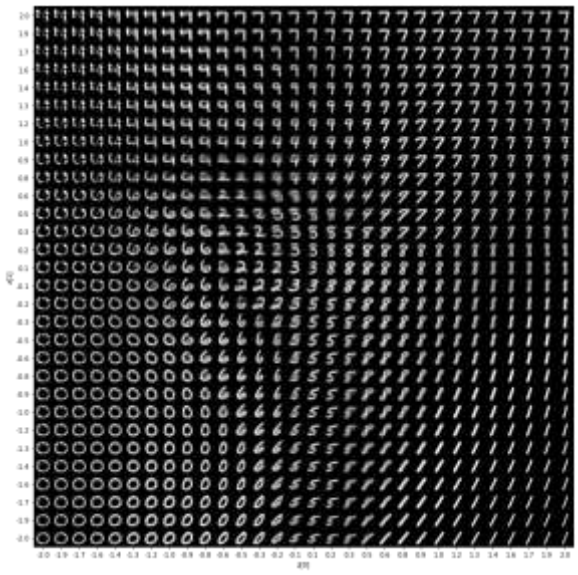

Figure 15 Epoch = 360

3 Conclusion

Both model VAE and GAN are successful in generating the digit's images. GAN is trained for 500 epochs and VAE for 360 epochs. We can observe as the number of epochs increases the images produced are getting clear, especially in GAN model. As for VAE the images are blurry, and not much improvement in the images produce in comparison to GAN, so stopped at 360 epochs.

Overall, tuning the models is a subtle task, required some research. There is not enough theory on how to fine tune the parameters to get better results, so the choice of models and training of the models would be good topics of research in near future.

The code and the more results can be found at the link: NikunjSarda/MachineLearning (github.com)

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672{2680. Curran Associates, Inc., 2014.
[2] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.
[3] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder, 2016.
[4] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2016.
[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.
[6] Min Lin. Softmax gan, 2017.