COP 6610(Machine Learning) Fall 2019

Project Report

Shashank Chandrashekar Murigappa

UFID: 37013155

smurigappa@ufl.edu

https://github.com/shashankcm95/Machine-Learning-Fall-2020-COP-6610

December 12, 2019

This report consists of the follwing three parts:

• Introduction and background

• Results of GAN(Generative Adversial Network) on MNIST dataset

• Results of VAE(Variational Auto Encoder) on MNIST dataset

1. Introduction and Background-

The GAN(Generative Adversial Network) model basically has a generator and a discriminator module where the Discriminator is responsible for verifying the authenticity of the image (whether the image has been sent by the user or the generator) and the Generator's job is to produce new images and send it to the Discriminator, as the discriminator keeps discarding these images, the generator learns gradually to create images that would be difficult for the discriminator to identify whether it was sent by the user of the generator itself.

The Variational Autoencoder Works as follows: First, an encoder network turns the input samples x into two parameters in a latent space, which we will note as z_mean and z_log_sigma. Then, we randomly sample similar points z from the latent normal distribution that is assumed to generate the data, via z = z_mean + exp(z_log_sigma) * epsilon, where epsilon is a random normal tensor. Finally, a decoder network maps these latent space points back to the original input data. The parameters of the model are trained via two loss functions: a reconstruction loss forcing the decoded samples to match the initial inputs, and the KL divergence between the learned latent distribution and the prior distribution, acting as a regularization term. We could get rid of this latter term entirely, although it does help in learning well-formed latent spaces and reducing overfitting to the training data.

2. Results of GAN(Generative Adversial Network) on MNIST dataset

The MNIST data in loaded by using Dataloader in PyTorch directly. Each element in "DataLoader" has an input image of 25 handwritten digits, and in each epoch, there are 600 batches. Apparently MNIST has 10 classes representing digits from 0 to 9, and the

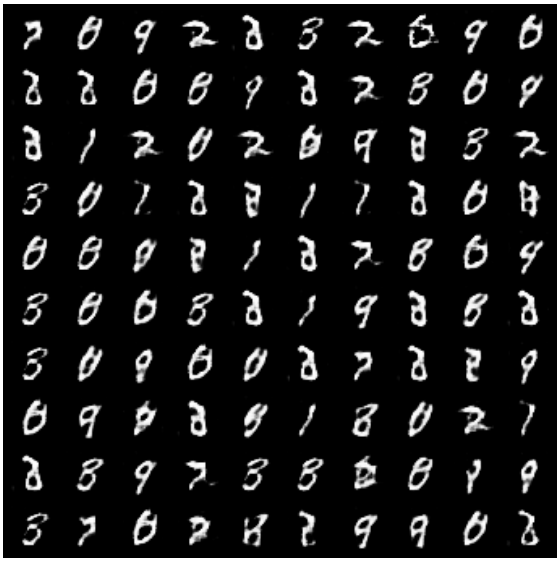
training/validation split is done automatically. The dataset will be downloaded automatically at run time.

Model description:

GAN for MNIST The GAN model used for MNIST is from the original Goodfellow directly. The generator consists of 5 linear layers, and between each leayers there's a LeakyReLU activation function. The discriminator consists of 3 linear layers, and also uses LeakyReLU as activation functions.

Results:

A few images have been attached from the result set. The full set of results are available on my GitHub repository(https://github.com/shashankcm95/Machine-Learning-Fall-2020-COP-6610) For MNIST results, output image is stored for every 5 epochs from 5 to 100. I do not know if it's the limited resources on my laptop or what, but it is taking more than 24 hours to run the 100 epochs, so I have only included these in the results. Figure 1 shows the output results for MNIST at step 600 for epochs, 5, 25, 45, 65, 85, 100. As we can see, the training result is pretty good, the output images look similar to the handwritten digits.



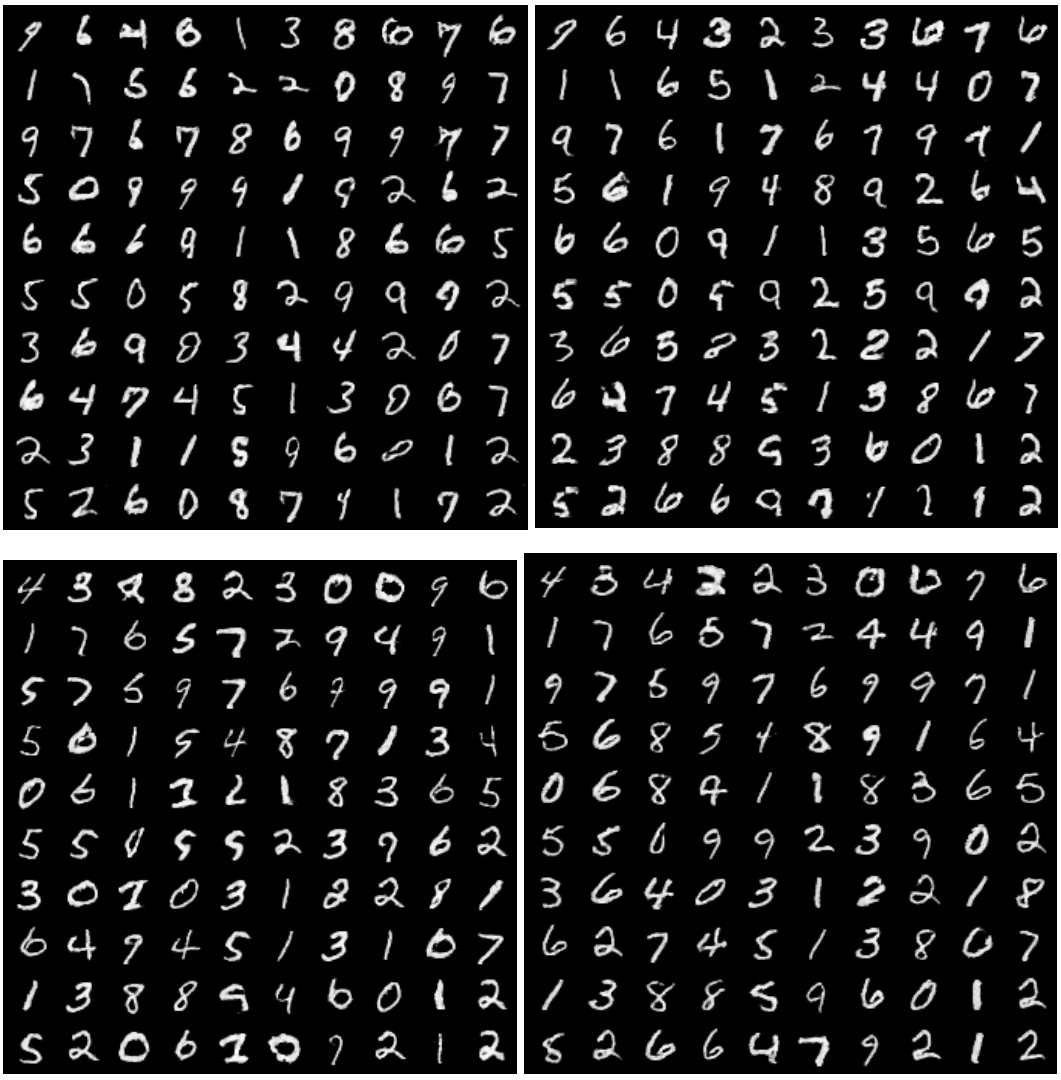


Figure 1: GAN Training result on MNIST at epoch values in the order 5, 25, 45, 65, 85, 100 respectively

3. Results of VAE(Variational Auto Encoder) on MNIST and face data

The input data on VAE is the same as GAN, which is shown in the previous section.

The encoder is two 4-liner layers, with first layer activated by ReLU. The two output from VAE(mu and logvar) share the first linear layer, then get separted since the second layer, and get reparameterzied to latent variable z in the end. The decoder is one linear layer with a ReLU.

My results are also attached on my GitHub. Results are shown as below: Figure 2 shows the output results for MNIST at epoch=0,10,20,30,40 and 50. The output images seems to

converge very fast, and after the first epoches, the quality of output images tend to be stable.

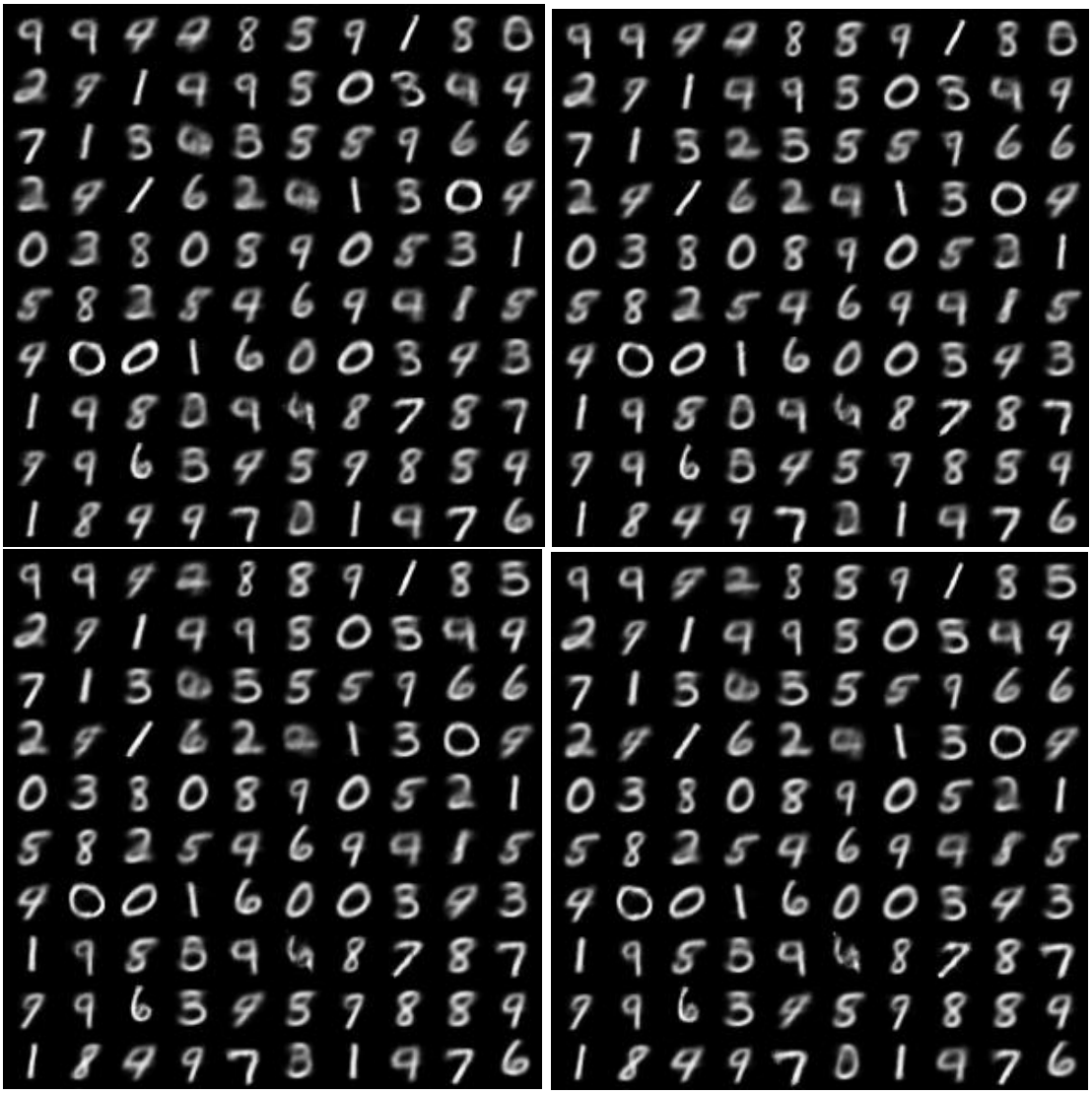


Figure 2: VAE Training result on MNIST after epoch in the order 5, 100, 200, 300, 400, 500 respectively.

4. Conclusion

To summarize, in this project, different types of ML models were used to generate fake images of hand-writing characters. It took more time to train the GAN model, but the results were closer to the real image when compared to the VAE model which took relatively lesser time. The model was run for 2000 epochs for the VAE model and 500 for GAN. Both of these can be used to generate images that look like the real image enough to fool anyone looking at it.