Generative Models: Comparing Generative Adversarial Networks and Variational Auto Encoders on MNIST dataset

Mansi Rangwani
Department of Information Technology
University of Mumbai
Chembur, Maharashtra
2019mansi.rangwani@ves.ac.in

Abstract

The aim of generative modeling is to train a model in such a way that it can learn the distribution of the training sample using unsupervised learning and produce the output with distribution having as much similarity with the training sample as possible. To achieve this, we make use of neural networks. This paper tries to compare two efficient models called Generative Adversarial networks (GAN) and Variational Auto Encoders (VAE) on MNIST dataset. After comparing the results of the two models, VAE generated much better quality images than GAN.

1 Introduction

Generative Adversarial Networks (GAN) and Variational Auto Encoders (VAE) are generative models that use unsupervised learning approach. GAN consists of a discriminator and a generator that can create new data that will look similar to the training dataset. For example, if the training dataset contains human faces then GAN will generate images of human faces. Similarly, VAE also a generative model, contains an encoder and a decoder. The aim of the autoencoder is to regularize the encodings so that its latent space has good properties to generate new data. These models are tuned as generative models because they learn the data distribution from the training dataset and can generate new data that looks similar to the training set. The paper makes use of MNIST(Modified National Institute of Standards and Technology) dataset for training these two models. The dataset is a computer vision dataset composed of handwritten digits where each image is a 28*28 pixel image.

2 GAN Architecture

GAN are deep learning based generative models that is used for unsupervised learning. It is a framework where 2 different neural networks compete against each other to create variations in the training data. GAN architecture consists of two sub-models, called Generator model and Discriminator model. Generator model creates new data from the training set and the discriminator model decides whether the data generated by the Generator model is real or fake. This is done using a binary classification problem with the help of sigmoid function that gives an output within the range of 0 and 1. 0 refers to fake and 1 refers to real data.

The generative model analyzes the data in such a way that after the training phase, the probability of the discriminator making a mistake maximizes. On the other hand, the discriminator calculates the probability if the data is coming from the generator or the real data.

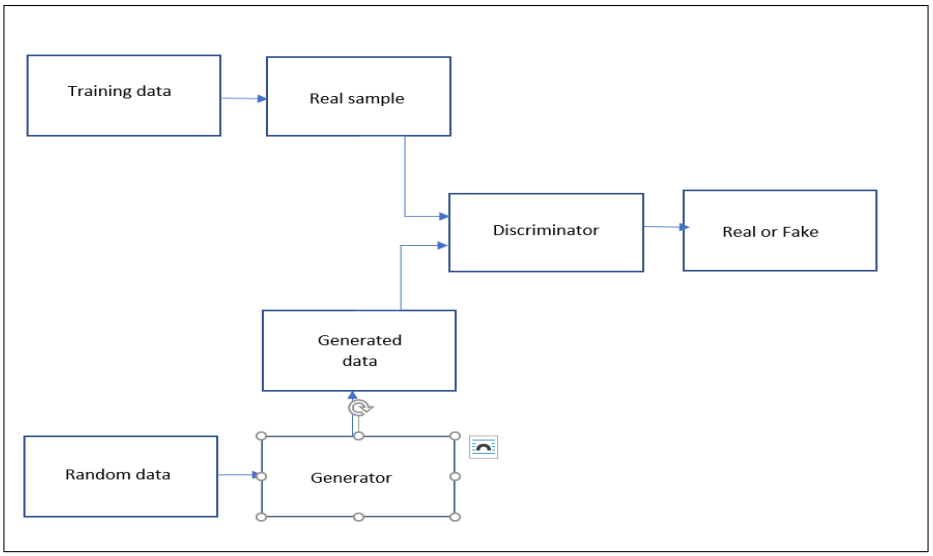


Figure 1: GAN Architecture

3 VAE Architecture

It consists of auto encoders that contains 2 sub modules called encoder and decoder. Encoder compresses down the higher dimensional input into a smaller dimensional representation called a bottleneck or continuous latent space, by using convolutional layers. The latent representation is stochastic i.e., they are parameters of probability distribution. The bottleneck vector contains 2 separate vectors called the mean vector and standard deviation vector. From this, the decoder then tries to reconstruct the input by using the de-convolutional layers. Bigger the size of the latent space, more optimized is the output of the decoder.

3.1 Figures

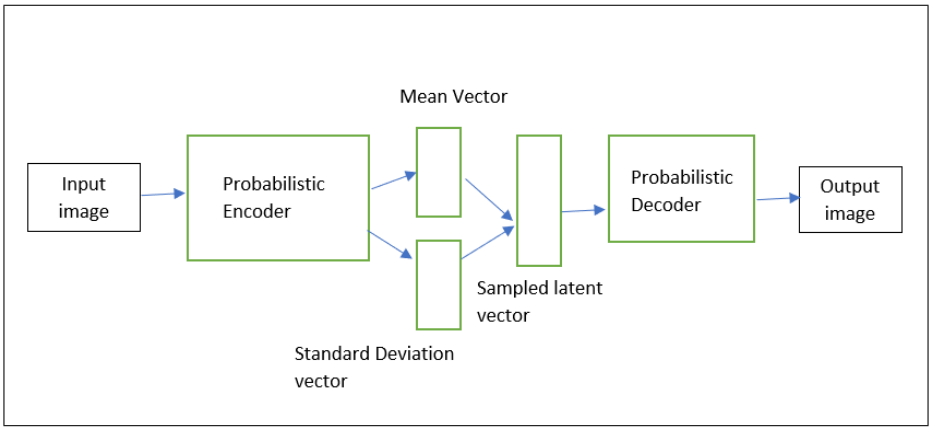


Figure 2: VAE Architecture

4 GAN Method

GAN is trained in 2 phases. The first phase includes training only the discriminator through forward pass. The training set for the generator is turned off and there is no backward propagation. In other words, the discriminator is trained with real data and fake data so that it can distinguish between them

correctly. The second phase includes training only the generator so that it can generate data samples that are as identical as possible to the real data.

4.1 Discriminator layer composition

I have implemented 4 layers. The first layer takes 784 neurons as input and gives 512 neurons as output. The second layer contains 512 neurons and gives 256 neurons as output. The following layer then takes 256 neurons and gives 128 as output. The following layers take 128 neurons and give output as 0(fake) or 1(real). Besides this, I have used a dropout rate of 0.3. The activation function used is leaky ReLU that allows for small negative slopes when the input is smaller than 0.

4.2 Generator layer composition

I have implemented 4 layers. The first layer takes 100 neurons as input and gives 128 neurons as output. The second layer contains 128 neurons and gives 256 neurons as output. The following layer then takes 256 neurons and gives 512 as output. The following layers take 512 neurons and gives the entire image of size 784 pixels as output. Besides this, I have used a dropout rate of 0.3. The activation function used is leaky ReLU that allows for small negative slopes when the input is smaller than 0.

The learning rate is 0.0002 and the loss function used is Binary Cross entropy combined with a Sigmoid layer in one class as it brings more numerical stability when compared to using Sigmoid layer followed by Binary Cross entropy.

5 VAE Method

In VAE's the training data (i.e., the MNIST images) is mapped to a latent space using neural networks. The latent space is the posterior distribution which is modelled as Gaussian distribution. So, as a result, the network gives two parameters as output i.e., mean and covariance of the distribution. Using this information, we sample the training data into the latent space and use it to generate data that is similar to training data.

5.1 Figures

$$-D_{KL}[q_\phi(z|x)||p_\theta(z)] + E_{q_\phi(z|x)}[log(p_\theta(x|z)]$$

Figure 3: Loss function

$$D_{KL}[q_\phi(z|x)||p_\theta(z)]$$

Figure 4: data reconstruction loss

$$q_\phi(z|x)$$

(a) output of the encoder i.e., mean and covariance

$$|p_\theta(z)$$

(b) mean and the standard deviation

KL divergence tells us how similar the two distributions are. If they are the same, it outputs 0. VAE loss function is the sum of reconstruction loss and the KL Divergence loss as demonstrated in Figure 3.

$E_{q_\phi(z|x)}[log(p_\theta(x|z)]$

(c) Regularization or the KL divergence

$log(p_\theta(x|z)$

(d) output of the decoder

5.2 Encoder layer composition

Encoder comprises of 3 layers. First layer contains 784 (28 * 28) nodes as input and gives 512 nodes as output, second layer contains 512 nodes as input and gives 256 nodes as output, third layer contains 256 input nodes and gives 2 output nodes. It makes use of ReLU activation function.

5.3 Decoder layer composition

Decoder also comprises of 3 layers. First layer contains 2 nodes as input and gives 256 nodes as output, second layer contains 256 nodes as input and gives 512 nodes as output, and third layer contains 512 input nodes and gives 784 output nodes. I have used ReLU and sigmoid activation function.

6 Observations

Problems faced while training GAN: -

- It is difficult to establish a stability between the generator and discriminator. If discriminator is too powerful then, it will classify most of the data as fake and if it is too lenient then, most of the generated data will be classified as real. So, there should be a balance between the two.
- It is difficult for GAN's to determine the position of the objects.
- Deciding the number of hidden layers and neurons in each layer for the generator as well as the discriminator.
- Deciding what activation function would work better. I finalized on leaky ReLU as it overcomes the problem of ReLU by allowing a small negative value when the input is negative.
- Deciding the most optimum learning rate for the model so that the discriminator and generator can learn at similar pace. If one of the two lags, then the other neural network is made to wait.
- Deciding what value to choose for the dropout layer that would prevent the model from overfitting the training data. In my case it came out to be 0.3.

Problems faced while training VAE: -

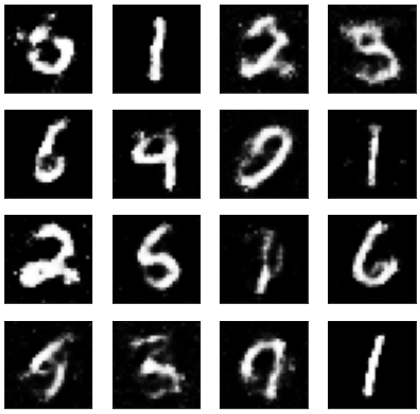
- Calculating the loss function was a bit difficult as there are no in built libraries for calculating the reconstruction loss and the KL divergence loss.
- Whenever the output of the encoder had to be changed, the input of the decoder also had to be modified.
- The amount of documentation available for using VAE's for generating images is quite less when compared to GAN.

7 Results

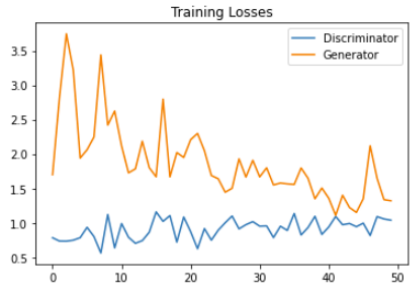
7.1 GAN Output

I have run GAN model for 50 epochs.

On inspecting the output images of GAN, some of the digits are recognizable and some are totally out of shape. The final generated images are as follows: -



(e) GAN output



(f) Generator and Discriminator loss convergence chart

7.2 VAE Output

I have run VAE for 51 epochs. From the output images of VAE, majority of the digits are recognizable.

Figure 5: VAE output

8 Conclusion

In VAE we make an assumption of optimizing the lower variational bounds whereas in GAN there are no such assumptions. One of the major challenges in GAN is that it is very difficult to converge the algorithm and keep it stable as both the generator and discriminator are constantly competing with each other. It turns out that VAE yields clear and better quality images than GAN as it is evident from visual inspection of their outputs.

9 References

[1] https://github.com/Sachit1137/Generative-Adversarial-Networks/blob/main/GAN.py

[2] https://github.com/Sachit1137/Variational-Auto-Encoders-VAE-/blob/main/VAE.py

[3] https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[4] https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[5] https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29