Comparing Generative Models: GANs and VAEs

Rishabh Khanna
UFID: 1339 9251
Graduate Student
CISE Department, University of Florida
`rishabh.khanna@ufl.edu`

Abstract

Generative Adversarial Networks and Variational Auto Encoders, both are examples of generative models. It is quite difficult to evaluate both in respect to each other as both use different principles of operation. This paper is an attempt to discuss about generative models and also compare the two generative models on MNIST data set. This paper explains their architectural principles as well as the problems both models pose.

1 Introduction

Generative models are a class of frameworks which aim at learning the distribution of a data set rather than learning its labels output. Generative models if trained well can be used used to produce samples of data set which mirror the distribution of data set on which they were trained. Interestingly the samples generated by the generative models do not belong to the data set on which they were trained or any sample which they have seen before. We can formally define generative models as a statistical model which is based on the joint probability distribution of two random variables A and B, where A is the feature variable and B is the label variable.

A simple analogy will be that generative models are required to learn a language and then speak that language as and when required. Applications of generative models are in Natural Language Processing(NLP), where the chat bots learn to speak a language and also create sentences on their own in order to cater to the needs of the situation. Amazon Alexa and Google Assistant can be considered to be using generative models to create more believable reply and it is evident that the training data set for such kind of models is not enough to know about all the user inputs. For example, Alexa can be trained to identify certain keywords like order, shop, ship, call but what if the pronunciation of the word is different or the user has some lisp and can't speak words correctly. In this scenario the model is required to understand the word and then reply correctly.[10]

When it comes to different categories of generative models[12], they can be categorized based on the their underlying statistical principle. Some of the categories are: -

1. Gaussian Mixture Model
2. Hidden Markov Model
3. Probabilistic context-free grammar
4. Bayesian network
5. Variational Autoencoder
6. Generative Adversarial Network
7. Flow-based generative model
8. Energy based model

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

With the upsurge of deep learning there is a new class of generative models which are based on deep learning models, these are referred to as Deep Generative Models(DGM).[1] These models use neural networks to large extent and also use statistically influenced loss functions and optimization techniques which make then a good candidate for generative models. In case of deep learning networks the generative models consider a smaller number of parameters as compared to the data set on which the models are trained. This enables the models to look for the distribution of the data rather than just learn the data set.

1.1 Generative Adversarial Network

The early deep generative models suffered form challenges and were unsuccessful as generative models. The major hurdle for DGNs was the hardship in approximating many unmanageable computations relating to probability which were result of maximum likelihood estimation(MLE) and similar approaches. In addition to this came the difficulty of utilizing the abilities of piece-wise linear units with respect to generative models. Addressing these concerns Generative Adversarial Networks were designed.[5] GANs have two entities, namely Generator(G) and Discriminator(D). Both get into a tussle to reduce their loss. Purpose of G is to generate new samples which resemble like a sample from the original data set. On the other hand D is tasked to beat G by classifying its data as fake. If D marks the output of G as fake then G corrects itself to align more closely to the original data set distribution. While D is already trained on the original data set to recognize the distribution. This makes D a good cop. G keeps on generating new samples until there comes a situation where G is able to generate samples which D is unable to classify as fake or unreal. This means that the samples generated by G truly represent the original data set distribution.

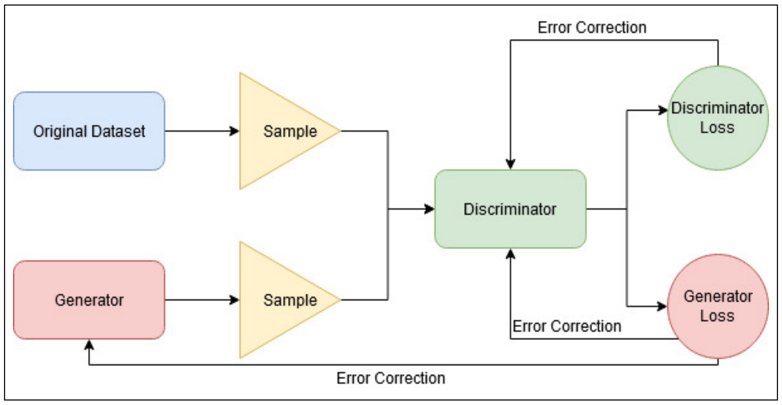


Figure 1: GAN

1.2 Variational Autoencoders

Autoencoders is a class of networks which compress the dimenionality of the original data and from compressed dimensionality original dimensionality can be decompressed. The result of decompression is supposed to achieve a quality equivalent to the original sample.[3] A generative model based on autoencoders is built using Variational Bayes Interference and these models tend to generate data in alignment to the probability distribution of the training data. This model is similar to a probabilistic decoder which uses neural network to find the conditional probability distribution function of the given input and then generate samples from that distribution. The other half of the VAE network can be thought of as a probabilistic encoder which encodes the given input into a latent vector[4]. In VAE the log-likelihood is considered to be the summation of all data and for individual point the log-likelihood can be expressed as -

$$log\, p_\theta(x_i) = D_{KL}(q_\theta(z|x_i)\, ||\, p_\theta(z|x_i)) + \mathcal{L}(\theta, \phi; x_i)$$

Here the first term is Kullback-Leibler divergence(KL) and the second half is lower bound on variation. These two help define the loss function for VAE network.[3]

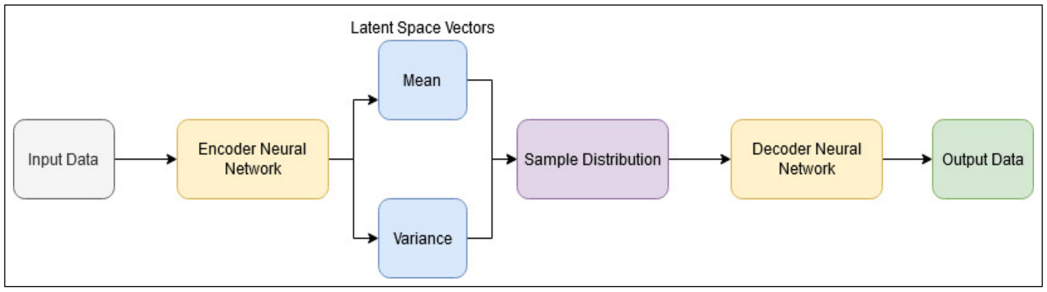


Figure 2: VAE

The comparison of GANs and VAEs has been up for debate from the time both have been used as generative models. There are people who feel GANs are superior than VAEs as VAEs do not have the ability to produce sharper images and their image quality is poor. However, when it comes to non-image data researchers argue that VAEs provide a latent distribution space which can be very useful in data imputation and completion and thus in such scenarios VAEs have a wider fan-base than GANs. This puts forth a challenge of comparing two models with the right metric. Researchers also argue that GANs suffer from the problem of vanishing gradient.[2] If the Discriminator network is too good than there are chances that Generator might suffer from vanishing gradient problem. There is very little known about such unstable nature of GANs.

Section 2 discusses the method employed in building the two models and training the. It also ocuses on the problems faced while developing both models. Section 3 presents the results of each model and the label predictions for each model's generated outputs. Section 4 is the conclusion and future possibilities of the paper.

2 Methodology

The comparison of two generative models was tough as both followed different structural models. A very precise care was taken when selecting all the technical pieces like libraries and functions which can be kept same in both models to reduce the chances of interference from external factors like library implementation, optimization function implementation, etc. For implementing both models Python was chosen and PyTorch, Pandas, NumPy were employed. Choosing these technologies was random and these have been accepted as state of art in building machine learning algorithms and in the field of data science. Neural networks were built using PyTorch. Both the models were built and test on the same machine with configuration as - Graphics Card - NVIDIA GEFORCE GTX 1660Ti, 6GB, RAM - 16 GB, Processor - Intel i7 9th Gen 9750H, 2.6 GHz. With graphic card capabilities running code with CUDA was possible rather than just running on CPU. MNIST was selected as the common data set. The data set is readily available and is widely used to benchmark models.

2.1 Generative Adversarial Network

The GAN model has two neural networks one for Generator and the other for Discriminator. The Generator model has the configuration as shown in Table 1. On the other hand Discriminator model has configuration as shown in Table 2. Algorithm 1 details out the GAN model algorithm employed. In each epoch real images were considered in batches. For each batch a fake image was generated and discriminator was trained with loss on real images as well as on loss on fake image. While Generator was only trained on loss by fake image. Binary Cross Entropy function of PyTorch was used as loss function to calculate both losses. As an optimization to Stochastic Gradient Descent Adam optimizer was used for both models with learning rate as .0002. Additionally, after each layer in Discriminator a dropout of 0.3 was done in order to enhance the capabilities of Discriminator.

Table 1: Generator Network

Layer	Weights	Biases	Output	Activation Function
Input	128	256	256	LeakyReLU
Hidden 1	256	512	512	LeakyReLU
Hidden 2	512	1024	1024	LeakyReLU
Output	1024	784	784	TanH

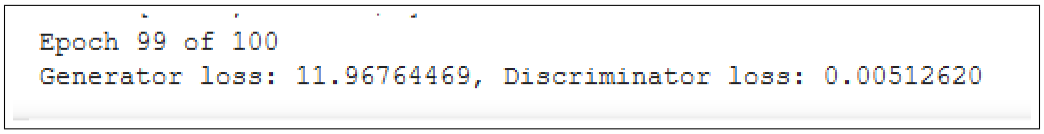
Table 2: Discriminator Network

Layer	Weights	Biases	Output	Activation Function
Input	784	1024	1024	LeakyReLU
Hidden 1	1024	512	512	LeakyReLU
Hidden 2	512	256	256	LeakyReLU
Output	256	1	1	Sigmoid

Algorithm 1: GAN

```
i = 0;
epochs = 200;
batchSize = 256 ;
while i<epochs do
    j = 0;
    while j<sizeOfDataSet do
        realImage = dataSet[j:j+batchSize];
        fakeImage = generate(randomValue);
        fakeLoss = calculateFakeLoss(fakeImage);
        realLoss = calculateRealLoss(realImage);
        TrainDiscriminator(fakeLoss + realLoss);
        TrainGenerator(fakeLoss);
        j+=batchSize;
    end
    i++;
end
```

GANs took longer to be trained and experimenting with GAN networks was tougher as compared to VAE simply because two networks need to be trained and both can have different configurations. Also, Generator many times would run into Vanishing Gradient problem as the loss would not increase nor decrease. This made Discriminator much efficient. Tuning hyper parameters for GANs was tougher even though the libraries provide very good implementation of the loss functions and gradient descent optimizers.[8]



```
Epoch 99 of 100
Generator loss: 11.96764469, Discriminator loss: 0.00512620
```

Figure 3: Generator stuck in vanishing gradient while Discriminator showing very low loss

2.2 Variational Autoencoders

Variation Autoencoder model has two halves the first half is the encoder and the network is described in Table 3. The encoder generates two outputs which are supposed to be mean and variance of the data set. This is also the latent vector for the VAE. From this sampling is done and the new sample taken from the latent vector is passed to the second half, which is the Decoder. The decoder takes in 2 inputs and generates an image of 784 size i.e. 28X28. Once the decoder gives its output the loss function which consists of two different losses, KL loss and variational lower bound, are calculated.

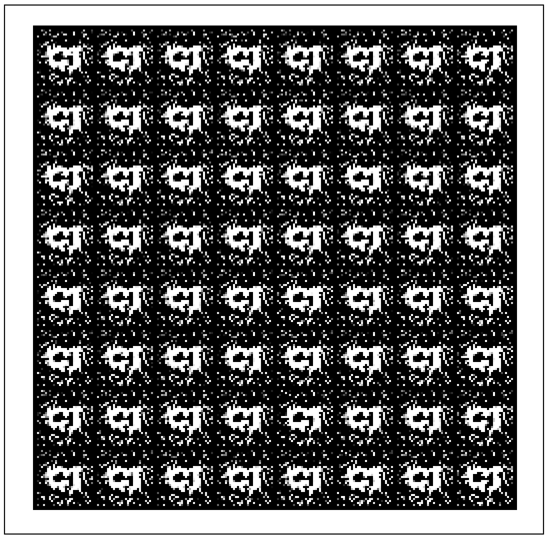


Figure 4: Output of GAN with vanishing gradient problem

The variational lower bound is calculated using Binary Cross Entropy and here also Adam optimizer is used for optimizing the gradient descent. Sum of these losses is backpropagated into the network and then the next cycle starts. Algorithm 2 describes the steps in order.

Table 3: Encoder Network

Layer	Weights	Biases	Output	Activation Function
Input	784	512	512	ReLU
Hidden 1	512	256	256	ReLU
Output 1	256	2	2	NA
Output 2	256	2	2	NA

Table 4: Decoder Network

Layer	Weights	Biases	Output	Activation Function
Input	2	256	256	ReLU
Hidden 1	256	512	512	ReLU
Output	512	784	784	Sigmoid

Algorithm 2: VAE

```
i = 0;
epochs = 5;
batchSize = 100 ;
while i<epochs do
    j = 0;
    while j<sizeOfDataSet do
        Image = dataSet[j:j+batchSize] ;
        OutputImage, KLD, reconstructionError = VAE(Image) ;
        TrainVAE(KLD + reconstructionError) ;
        j+=batchSize;
    end
    i++;
end
```

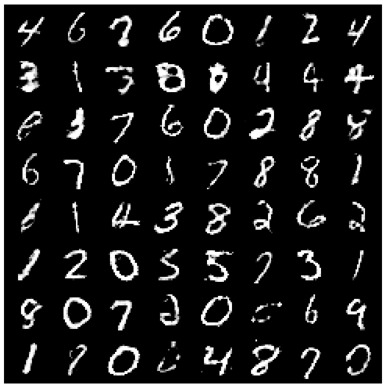
While building model for VAE it was hard to turn the mathematical equation into loss function. This couldn't be hit and try as the loss had specific meaning in this and just Binary Cross Entropy loss function could not be used here like GANs. For this a thorough research of VAE literature and various blogs were done and then a loss function was built.[6, 7, 3, 4] Listing 1 describes the loss function. F is the tensorflow library identifier and mu is mean while log_var refers to variance. torch identifier is used for PyTorch library.

Listing 1: Loss function used for VAE

```
def loss_function(recon_x, x, mu, log_var):
    BCE = F.binary_cross_entropy(recon_x,x.view(-1,784),reduction='sum')
    KLD = -0.5 * torch.sum(1 + log_var - mu.pow(2) - log_var.exp())
    return BCE + KLD
```

3 Results

Both models were trained on MNIST and then random noise was used to generate MNIST data. Different combinations of activation functions, layer configurations, loss function implementations were tried in order to achieve visually good looking output. Using any loss as the criterion to decide model wasn't feasible as the loss values can be ambiguous in the starting. For example in case of GANs, Discriminator is also not trained well enough and neither is Generator and hence their losses are high and equal but this can't be treated as a stopping point. There were also stages where in the loss percentage was low for both networks however the output wasn't satisfactory. The output was just a scattered image of random points. Hence, by repeated trial and error GANs and VAEs were trained to provide visually acceptable results.


(a) GAN generated MNIST

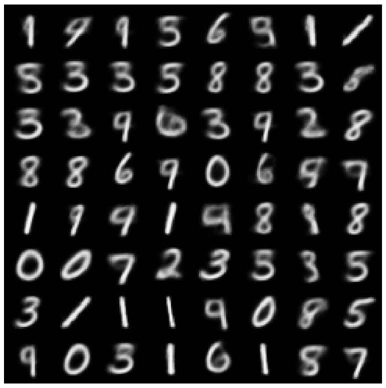

(b) VAE generated MNIST

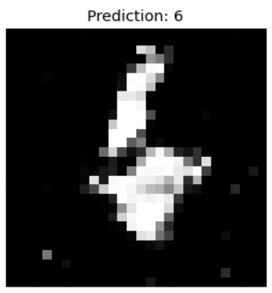
Figure 5: MNIST generated by GAN and VAE

3.1 Generative Adversarial Network

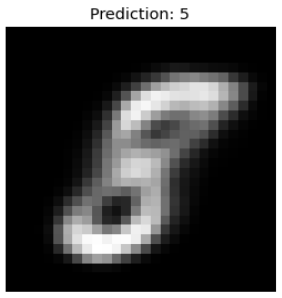
GANs provide very sharp images of numbers, Figure 5 (a). However there are few number which are still malformed as shown in Figure 6 (a). Majority of the numbers were predicted correctly by the MNIST predictor model as shown in Figure 7. However, this is just a cursory test and cannot be taken as a judgement metric for comparing GAN and VAE.

3.2 Variational Autoencoders

As seen in Figure 5 (b) the digits are easy to recognize except few. However, the images appear little blurry or smudgy. This also means that few numbers can be misunderstood easily such as shown in figure 6 (b). Prediction by MNIST prediction algorithm trained on MNIST is stated in Figure 8 for few outputs of VAE.



(a) GAN generated MNIST with predicted value as 6



(b) VAE generated MNIST with predicted value as 5

Figure 6: Ambiguous generation

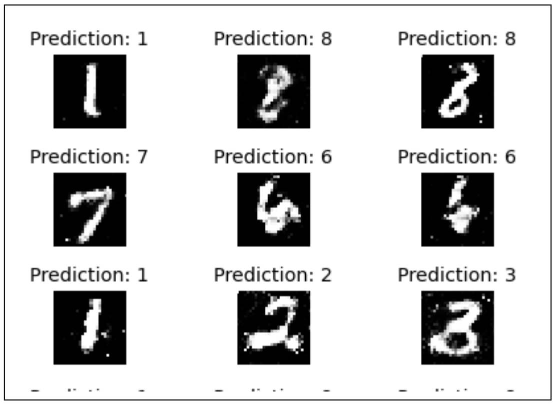


Figure 7: GAN generated MNIST along with predicted labels

4 Conclusion

The study on comparing two generative models GANs and VAEs was successfully completed on MNIST data set. Results form both the models were some what satisfactory. Through eyeballing it was found that images of MNIST generated by GANs is much more sharper and realistic. On the other hand VAEs produced images even though distinguishable but appear to be blurry. Another approach of comparison could have been the error rate but as both depend on different error calculations, this means that in GANs Generator and Discriminator both compete with each other and both try to reach a Nash Equilibrium point while this is not the case in VAEs where KL divergence is tried to minimized and log-likelihood is maximized. A label predictor model was trained on MNIST data and then data generated by VAE and GAN models was fed into that but that technique wasn't much use as the label for each generated image was missing and there were times when the generated image looked like a zero or eight and the label predictor model predicted 8. This result can neither be rejected nor accepted. On the other hand the figures which were clear to the eye had quite similar prediction accuracy for both the models. Hence, this technique was not very helpful and it is also data set dependent. Reading different literature did not yield much insight on the comparison techniques used by others. However, the literature did yield the position of each model in solving various problems. It is accepted that VAEs are probabilistic graphical models which aim at modeling the latent space and then marginalizing out some variables which is included in the modelling itself. They generate good results where latent space is of high importance. On the other hand, GANs are powerful and are used mostly with image data as they produce better results. A combined approach of GAN and VAE could lead to interesting insights and with recent studies a hybrid model have been proposed. This Hybrid model is based on intuition that Generator of GAN is actually a VAE and uses latent space to generate new data.[11] Further enhancement in the current study can be using CNNs instead of usual neural networks in the generative models. DCGAN is one example of this.[9]

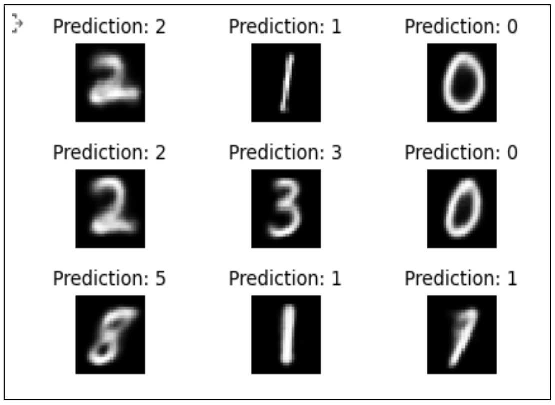


Figure 8: VAE generated MNIST along with predicted labels

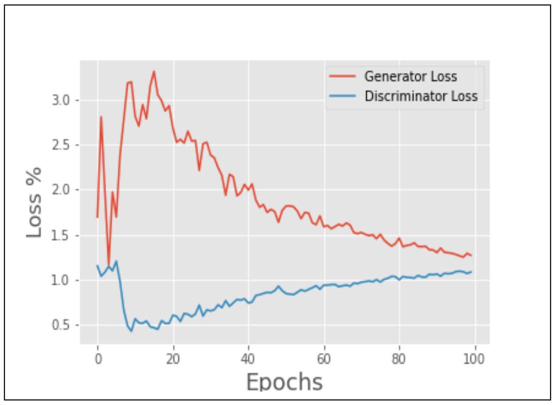


Figure 9: Generator loss vs Discriminator loss for GAN

References

[1] Open AI. *Generateive Model*. `https://openai.com/blog/generative-models/`.

[2] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2020.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[6] Anand Krishnamoorthy. Pytorch vae. `https://github.com/AntixK/PyTorch-VAE.git`, 2020.

[7] Allen Lee. pytorch-mnist-vae. `https://github.com/lyeoni/pytorch-mnist-VAE.git`, 2018.

[8] Erik Linder-Norén. Pytorch-gan. `https://github.com/eriklindernoren/PyTorch-GAN.git`, 2018.

[9] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

[10] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda

Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. Conversational ai: The science behind the alexa prize. 2018.

[11] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks, 2017.

[12] WikiPedia. *Generateive Model*. `https://en.wikipedia.org/wiki/Generative_model`.