Comparative Analysis of GAN and VAE

Prajan Tikayyolla
Department of Computer & Information Science
University of Florida
prajantikayyolla@ufl.edu

December 11, 2020

Abstract

This paper presents a comparative analysis between two Generative Models i.e., GAN and VAE. Firstly, studied the architectural behaviour of both the models in depth and compared the difference between them. Next, the behavioural change of both GAN and VAE with model parameters have been studied and obtained the best fitting parameters for MNIST dataset. Finally, contrasted the results of both models against MNIST dataset and found that GAN provided sharp images whereas VAE gave blurry images and the amount of time it took to train the model was less for VAE compared to GAN.

1 Motivation for GAN and VAE to be Generative Model

Supervised learning is a technique where the model tries to predict the output by learning from input data which contains labels. In supervised learning the model which tries to predict the class of the input is called classification and it is traditionally referred to as discriminative modeling. The unsupervised learning is the one in which the model tries to find patterns in the input data and the input data does not have any labels. This process of learning the distribution of input data make the unsupervised learning to generate new data from the learnt distribution and these models of generating new instances of data from the distributions is called the generative modelling.

Generator Adversarial Network are an intuitive way of training a generative model, where it possess the unsupervised learning problem into supervised learning problem with two models the generator which is used to generate new instances of the data and the discriminator which tries to classify the fake data from the real. This model is trained based on min max loss, the generator tries to fool the discriminator and discriminator tries to be smart enough to correctly classify the data.

Variational Auto Encoders provides probability distribution of the latent state variable instead of resulting single valued latent state representation as the auto encoders. Variational auto Encoders consists of two parameterized models: the encoder or the recognition model and the decoder or the generative model. The recognition model provides an approximation to its posterior over latent state variables to the generative model. The approximation model is indeed approximate inverse of the generative model.

2 Architecture Review

2.1 GAN Architecture

Generator Adversarial Network consists of two parts, generator, and the discriminator. The generator takes in random noise and tries to generate new images from the same problem domain and these generated data is fed as negative examples to the discriminator. The discriminator then tries to classify the fake and real data.

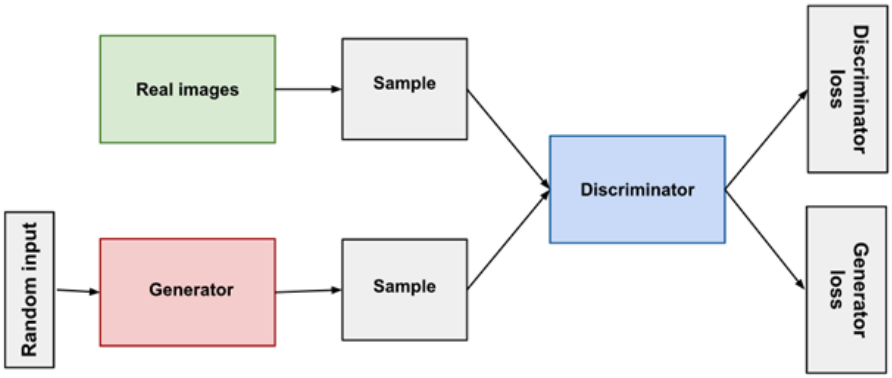

Figure 1: GAN Architecture.

The discriminator just tries to classify the real and fake data. The real images are fed as positive examples and generated images are fed as negative examples to discriminator while training the discriminator. The discriminator is connected to both the losses but during its training it ignores the generator loss. The discriminator is penalized for wrong classification i.e., by back propagation from discriminator loss it tries to update its weights.

Generator training is tightly coupled with the discriminator as it tries to make sure that discriminator fails to correctly classify the data. As we can see from above figure that random noise which is taken from unit Gaussian distribution is fed to generator to generate new instance and is then fed to discriminator and then it produces the output we are trying to effect. The generator loss penalizes the generator for allowing the discriminator to correctly classify the input. This loss helps in adjusting the weights of only generator and not discriminator. If

the discriminator also tries to change its weight while training the generator then it leads to intractable solution.

The training of both these models happen in alternating fashion i.e., firsts discriminator tries to train its model keeping generator constant and then after some epoch then generator tries to train keeping discriminator constant and this continues until the discriminator reaches a saturation where it tries to predict the data randomly i.e., the discriminator has only 50% probability to classify correctly and this is where the GAN reaches convergence and generator should stop training or else it might decrease its performance. The loss function for GAN is given below. The generator tries to minimize it and the discriminator tries to maximize it and it is called as Minimax Loss

$$E_x[log(D(x))] + E_z[log(1 - D(G(z))] \tag{1}$$

The main motivation for the variational auto encoders is that the decoder part of it can be used as generative model. By sampling from the latent state representation, the decoder can be used to generator to produce the new data like the prior distribution. The sample from which they are drawn the prior distribution is assumed to be unit Gaussian distribution.

2.2 VAE Architecture

The main motivation for the variational auto encoders is that the decoder part of it can be used as generative model. By sampling from the latent state representation, the decoder can be used to generator to produce the new data like the prior distribution. The sample from which they are drawn the prior distribution is assumed to be unit gaussian distribution.

The VAE takes the input data and process it through the encoder network to bring down the actual representation to some learnt latent state variable. Then samples are drawn from the latent state distribution and provided to the decoder model to generate the original data.

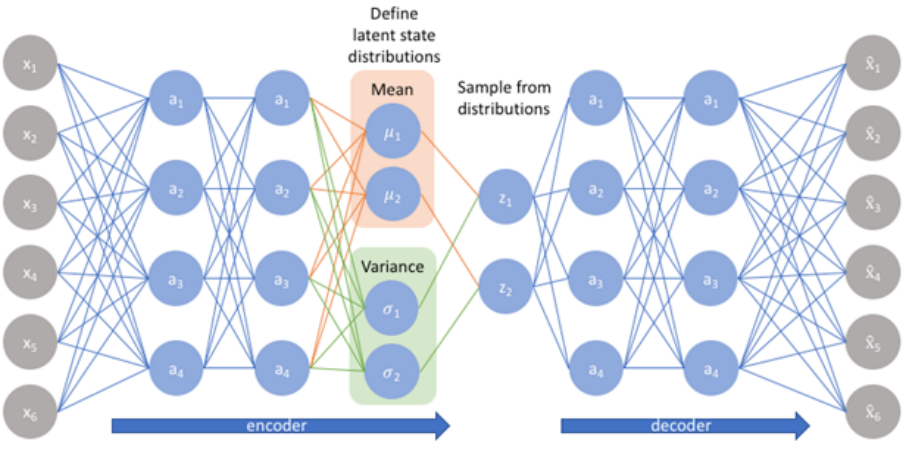

Figure 2: VAE Architecture.

The input data which is the x vector must have been generated from some hidden variable z so we need to learn the hidden variable z i.e., p(z|x) but computing this is difficult because p(x) is intractable distribution. One way to overcome is by variational inference i.e., we try to approximate p(z|x) with other distribution q(z|x) which we will make it tractable distribution by setting appropriate parameters. Then we can use this approximate inference on intractable distribution p(x). Now the main problem is that we need to approximate q(z|x) to be as similar as possible to p(z|x) but how do we ensure that it is similar, we can use KL divergence which is the measure of variability between two distributions. So, minimizing the KL divergence between these two distributions make sure that we have a good approximation by which we can learn the hidden distribution which generates the data.

$$minKL(q(z|x)||p(z|x)) \tag{2}$$

Minimizing KL divergence is equal to maximizing

$$E_(q(z|x))logp(x|z) - KL(q(z|x)||p(z|x)) \tag{3}$$

The first term is the expression is talking about the reconstruction likelihood and the second term infers us about the similarity of two distributions. Now that we maximize the above equation we have a good approximation of the prior distribution so we use this distribution to study the latent state variables z and then generate the original data back by sampling from the latent state distribution.

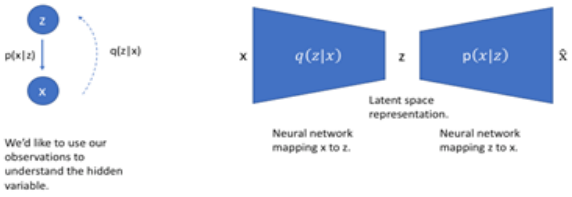

Figure 3: Encoder & Decoder block.

Thus, our loss function for variational autoencoders can be written as

$$L(x, x') + \sum_j KL(q_j(z|x)||p(z|x)) \tag{4}$$

$L(x, x')$ represents reconstruction error. This can be calculated using Binary Cross Entropy.
$\sum_j KL(q_j(z|x)||p(z|x))$ represents the likelihood of the distributions.
j represents the dimension of the latent state.
The sampling from latent state representation uses the reparameterization technique, where we randomly sample form unit gaussian and the move it by mean and scale with the log variance of the latent state representation. The reparameterization is used to because we need to learn the features while doing back propagation.

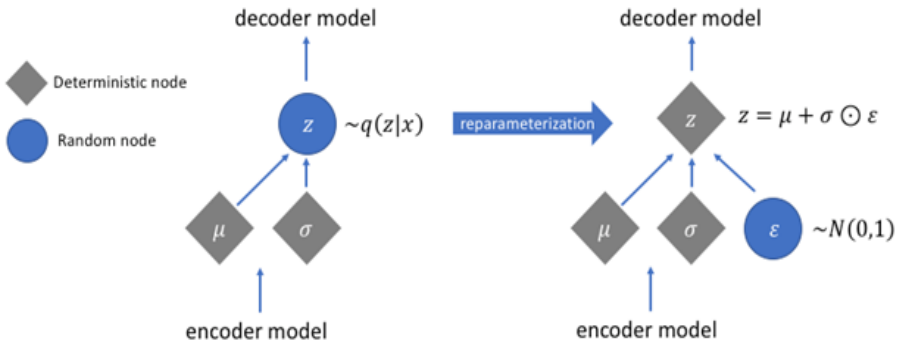

Figure 4: Sampling.

3 Architectural Difference between GAN & VAE

The generative models are of two types explicit density model and implicit density mode. The explicit density model defines a density function i.e., it assumes about the prior distribution which in our case is the VAE. The implicit models directly generate new data by using stochastic procedures which in our case is the GAN. Both the models primarily differ in the way they train their models. The generator and discriminator part of GAN's tries to learn in an alternating fashion i.e., first for some time the discriminator trains and then generator trains and these to keep training in alternating fashion. Whereas the encoder and decoder part of the VAE's train themselves together i.e., with every batch weights of both networks are adjusted. The plain vanilla version of VAE produces blurry images whereas the GAN produces crisp images. In addition, GAN's vanilla implementation faces mode collapse problem where generator does not produce new data but this in fact can be solved using Wasserstein loss. The one advantage of VAE over GAN is that it tries to learn the distribution which can used in addition to the generative model.

4 Training

4.1 VAE Training

The parameters that I have considered the most while training the VAE model were batch size, number of epochs, number of convolutional layers and optimizer. The loss function was straight forward which included KL divergence loss which tells how similar the approximated distribution was like the true distribution and reconstruction error for which I have used the binary cross entropy. Then tried with batch, mini batch mode on batch size on found that batch needs huge memory to load the entire data to memory and the number of convergence steps required comparatively less but the time it took for each step was more. Whereas in mini batch took batch size smaller than total dataset size and found

it more steps and it was noisy but was fast as time required to process each batch was less and with each batch weights were updated. Then compared the performance of model based on number of epochs and inferred that after certain number of epochs the model almost performed same. Adding more convolutional layers did not improve the model performance. The optimizer used was Adam.

Figure 5: Epoch 10

Figure 6: Epoch 20

Figure 7: Epoch 30

Figure 8: Epoch 40

Loss(–>)	2 Hidden Layers	3 Hidden Layers	4 Hidden Layers
Epoch Number			
10	143.9419	141.4297	140.8994
20	141.5191	139.4731	138.7821
30	140.0324	137.3182	144.2723
40	140.322	136.7412	137.5407
50	138.811	135.9628	140.8052
60	139.8912	138.0652	140.9192
70	139.1503	136.5565	144.1735
80	137.9177	136.0789	143.6392
90	138.1114	136.4146	139.5986
100	138.4062	136.4544	140.4299

Table 1: Loss with respect to number of hidden layers and epochs.

This table shows loss for the model with configuration on number of hidden layers and number of epochs and the batch size considered was mini batch with batch size of 100. As we can see after 50 epoch the model almost performed the same and the number of hidden layers did not affect the model.

4.2 GAN Training

Training the GAN model included starting from normalizing the data and as training the GAN model was taking long time. I worked with mini batch as batch mode will consume more time to make a progress in the direction of optimization. The model suffered from stability initially because of use of ReLU function along with convolutional layers but it was resolved when I went on with Leaky ReLU. The optimizer used was Adam with learning rate of 0.002 and most of my model settings have been implemented by referring to GAN hacks from github.

Loss(–>)	Discriminator Loss	Generator Loss
Epoch Number		
1	1.018	2.863
20	0.864	1.690
40	1.080	1.250
60	1.192	1.037
80	1.243	0.949
100	1.259	0.922
120	1.263	0.921
140	1.280	0.885
160	1.287	0.874
180	1.291	0.860
200	1.286	0.870

Table 2: Loss of discriminator and generator with number of epochs.

The generator initially performed poorly whereas the discriminator performed good in starting stage of training. As the number of epochs increased the

discriminator started losing its performance while generator gained meaning the images generated by generator where so real that discriminator was failing to detect fake images from the real images. Used dropout layers to add regularization effect to not overfit the model.

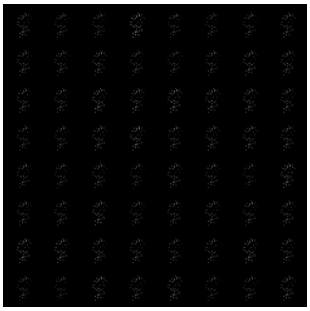

Figure 9: Epoch 1

Figure 10: Epoch 40

Figure 11: Epoch 80

Figure 12: Epoch 120

Figure 13: Epoch 160

Figure 14: Epoch 200

5 Comparison of results from GAN & VAE

Figure 15: GAN output

Figure 16: VAE output

The main difference with both models lies in their learning process. VAE tries to solve semi supervised problem whereas GAN solves unsupervised problem. The stability of GAN model makes user to select it but at the cost of training time. Comparing the images generated by models we can draw some conclusions i.e., the GAN produced crisp and sharp images whereas VAE produced blurry images. This contrast is with the MNIST data set only as we have trained the model only on MNIST data set because of lack of GPU. The amount of time to train was very less for VAE as compared to huge training time for GAN. Therefore using VAE we can get results in less time but at the cost of clarity as opposed to GAN's. Both have their own advantages and disadvantages. If quality is a constraint then GAN is better and training time is of high priority then choosing VAE is wise but at cost of clarity.

References

.

[1] Goodfellow, I. J., "Generative Adversarial Networks", arXiv e-prints, 2014.

[2] Kingma, D. P. and Welling, M., "An Introduction to Variational Autoencoders", arXiv e-prints, 2019.

[3] Goodfellow, I., "NIPS 2016 Tutorial: Generative Adversarial Networks", arXiv e-prints, 2016.

[4] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org.

[5] Mi, L., Shen, M., and Zhang, J., "A Probe Towards Understanding GAN and VAE Models", arXiv e-prints, 2018.

[6] Diederik P. Kingma, Max Welling. Auto-Encoding Variational Bayes, Machine Learning Group, Universiteit van Amsterdam(2014).

[7] Kingma, Diederik P and Welling, Max. Autoencoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[8] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[9] https://github.com/soumith/ganhacks

[10] I. Goodfellow, J.Pouget-Abadie, M.Mirza, B.Xu, D.Warde-Farley, S. Ozair, A.Courville, and Y.Bengio."Generative adversarial nets ". In Advances in neural information processing systems, pages 26722680, 2014.

[11] Shane Barratt, Rishi Sharma. A Note on the Inception Score, arXiv:1801.01973v2 [stat.ML](2018).