International Journal of Computer Applications (0975 - 8887)
Volume 182 - No.22, October 2018

Comparative Study of GAN and VAE

Jaydeep T. Chauhan
Student, B.Tech(CE)
DDIT, Nadiad
Gujarat, India

ABSTRACT

Generative models are very popular in a field of unsupervised learning.They are tremendously successful to learn underlying data distribution of training data and generate a new data with some variations.This paper presents a detailed study of generative models and how they differ from traditional discriminative models.The paper more focus on two most popular generative models such as Variational Autoencoder(VAE) and Generative Adversarial Network(GAN).The paper includes working of these generative models, their architecture and an experiment is conducted to generate images using very popular MNIST data set.The comparison between these two models and their advantages and disadvantages are presented based on an experiment.At last, some solutions are presented to further improve these models.

Keywords

Generative models, Unsupervised learning, Generative Adversarial Network, Variational Autoencoder, Machine Learning.

1. INTRODUCTION

There are two types of model in machine learning;discriminative models and generative models. Discriminative models are mostly used in supervised learning,where there is some dependency between variables y and x and main task is to predict y based on x.This dependency or mapping function between x and y can be learned from training data.This can be represented as:

$$f(x) = argmaxP(y/x). \quad (1)$$

Which chooses the class y that is most likely considering x.So in discriminative models are essentially learned to find decision boundary between different classes and it will predict y based on x.Different algorithms like Decision tree, Naive Bayes, SVM(Support Vector Machine), Artificial Neural Network are fall into this category.Here It didn't care about how the data is generated.Generative models are actually finding data distribution using joint probability:

$$f(x) = argmaxP(y/x) * P(y). \quad (2)$$

So this basically learn to give score to the configuration determined by x and y together and find y for a new x, so the joint probability became maximum.Algorithms like Generative Adversarial Network, Boltzmann machine, Variational autoencoder, auto-regressive networks are widely fall into this category.

2. CLASSIFICATION OF GENERATIVE MODELS

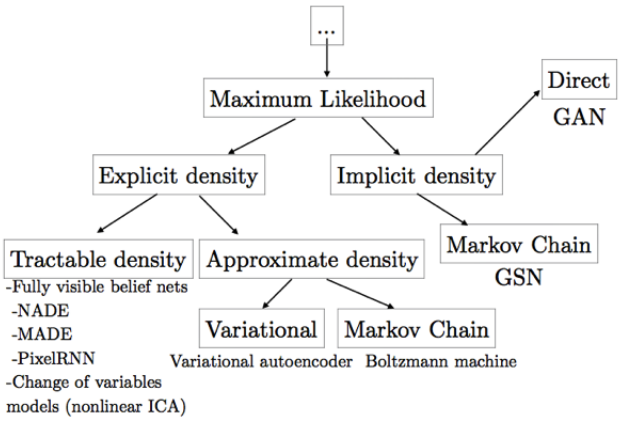

Fig. 1. **Generative Models** [2]

All types of generative models aim at learning the true data distribution of the training data set and generate new data samples with some variation.Classification of generative models can be done based on maximum likelihood estimation.It helps to estimate parameters of a model that best fits the probabilistic density function of training data.There are two types of generative models:implicit and explicit.In explicit model, the task is to learn probability distribution underlying the data explicitly.explicit model assume some prior distribution about some data and directly find density function in form of likelihood using optimization algorithm.Explicit models falls into two category:tractable and approximate.Tractable explicit models are computationally tractable.Popular models like pixelrnn, pixelcnn falls into this category.Where else in approximate explicit models are not computationally tractable.so here various methods of approximations are used like variational methods, stochastic approximations etc to define density function.Variational autoencoder and Markov chain falls into this category.Implicit generative models learn data distribution without explicitly define a density function.It is an implicit because it can only generate samples and cannot evaluate likelihood of data distribution.

3. WORKING OF GENERATIVE MODELS

3.1 Variational Autoencoder

Variational autoencoder[3] is a very popular generative model.It is a combination of two neural network.First network is an encoder network, which takes a data as an input and learn some hidden latent representation of a data.Encoder network convert input data into an encoding vector.Each dimension of an encoding vector represent some feature about a data and is a single value.For example, an encoder network of a generative model that generates an image of human face, will learn features of a human face like smile, hair color, skin tone etc and represent it in a form of encoding vector with some single value for each feature.Second network is a decoder network, which will take input an encoding vector and learn to generate an original data as an output.Variational autoencoder differs from a traditional autoencoder by instead of learning fix latent representation, it will learn probabilistic distribution for each latent feature of a data.

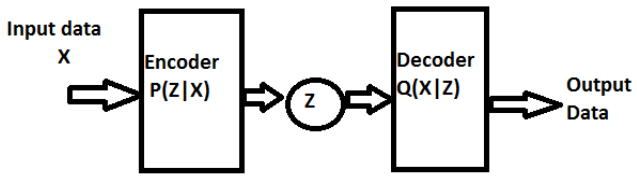

Fig. 2. **Architecture of Varitional Autoencoder**

The goal of Encoder network is to infer hidden latent variable Z from X as shown in figure 2.Encoder network can infer by estimating probability of $P(Z \mid X)$.

$$P(Z \mid X) = \frac{P(X \mid Z) * P(Z)}{P(X)}. \tag{3}$$

Now to compute $P(X)$ is a computationally intractable.

$$P(X) = \int P(X \mid Z)P(Z)\mathrm{d}z. \tag{4}$$

To solve this problem another technique called variational inference is used.Variational inference is used to estimate posterior $p(z \mid x)$, by defining a new distribution $q(x \mid z)$ such that it is a computationally tractable distribution.How to make sure that the parameters of a new distribution $q(x \mid z)$ are similar to the original distribution $p(z \mid x)$? Kullback-Leibler(KL) divergence is a very popular variational inference technique, which measure the difference between two probability distribution.

$$KL[Q(Z \mid X) \mid\mid P(Z \mid X)] = E[\log Q(Z \mid X) - \log P(Z \mid X)]. \tag{5}$$

Now to get the similar distribution, the KL divergence between those two distribution should be minimum.So the cost function contains two terms, first term represent the reconstruction loss which ensure that it is able to distinguish different classes uniquely and second term KL divergence that ensure that our new distribution q is similar to the prior distribution p.

$$E_Q(Z \mid X) \log P(X \mid Z) - KL[Q(Z \mid X) \mid\mid P(Z)]. \tag{6}$$

After in training phase, Model parameters can be learned for both the encoder and decoder network by applying optimization algorithm like stochastic gradient descent on above loss function.

An experiment is carried out to evaluate performance of this model.Variational autoencoder is trained on MNIST dataset.It is a very popular dataset.Pytorch and tensorflow frameworks are used to implement this model.Encoder is implemented as a convolutional neural network.Encoder contains one input layer, four hidden layers which performs convolution operations and two fully connected layers.Decoder contains two fully connected layers, four hidden layers which try to reconstruct output image and one output layer.Adam optimizer is used to train the networks and learning rate is set to 0.01.Batch normalization technique is used to speed up the learning process.The result of different epochs are shown below.

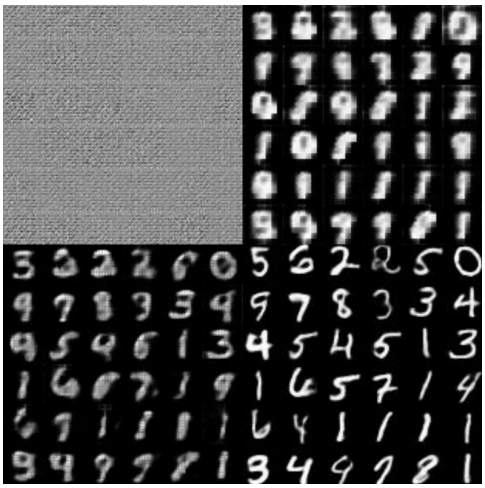

Fig. 3. **Visualization of generated images by VAE.Above images are sampled at four different epochs .(i)first epoch (ii) second epoch (iii) fifth epoch (iv) hundred epoch**

As shown in the above figure, Image quality is improve as number of epoch increases.In experiment, If model is exposed to only KL loss then it is shown that most of that data distribute around the center of the latent space.Due to the dense data distribution, it is difficult for decoder to decode any meaningful from latent space.Contrarily, if KL divergence completely removed from the model then data distribution become less smooth and more discrete.As a result, there is a less variation in an output data.Decoder perfectly able to reconstruct an original input data.It is very important to optimize both the loss in a model so that decoder can able to decode a sample perfectly and learn smooth data distribution.

Applications:-

—New images can be generated by replacing standard encoder network with convolutional and decoder network with deconvolutional network.

—By adding another sequence to sequence architecture like LSTM to the model, More creative things can be generted like synthetic text and music such as Google's Magenta MusicVAE[4].

—Google's Magenta sketchrnn is also very popular application of VAE[5].

3.2 Generative Adversarial Network

Generative Adversarial Network[1] has shown a very good results in many tasks to generate images, music, text etc.It takes a game theoretic approach unlike other conventional generative models, which learns to approximate complex density functions.It learns to generate by two player game between training.The two players are generator network and discriminator network.Generator network tries to generate data which is indistinguishable from training data distribution. The discriminator receives the sample and determine whether it is real or fake.The generator is trained to generate samples that fool the discriminator network, and discriminator is trained to make accurate predictions.

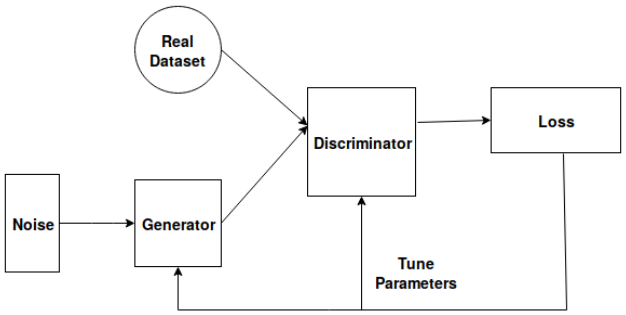

Fig. 4. **Architecture of Generative Adversarial Network.**

Generator is a neural network with parameter θ_1 and it's role is mapping some random input noise ***z*** to the data space.Discriminator is also a neural network that takes ***x*** as a input and parameter θ_2 and it outputs a probability that the data came from real dataset in a range of (0,1).Both this networks have their own cost functions and try to optimize their parameters.Generator's parameters are optimized to maximum the probability that any fake image is classified to belonging to real dataset.In mathematical terms, generator is a differential function *G* and discriminator is a differential function *D*.Generator tries to maximize the function *D(G(z))* or minimize the term $log(1 - D(G(z)))$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim pdata(x)}\left[\log D(X)\right] + \mathbb{E}_{z\sim pz(z)}\left[\log(1 - D(G(z)))\right] \quad (7)$$

Discriminator tries to maximize the whole above cost function by maximizing $logD(x)$ and maximizing $log(1 - D(G(z)))$ and generator tries to minimize only $log(1 - D(G(z)))$ in cost function.This is called minimax game because optimization can be achieved by minimize the inner loop and maximize the outer loop.Author[1] presented an algorithm to find convergence of this game.In training, there are alternate sequence between optimizing discriminator in k steps and optimizing generator in one step.Two mini batches are selected, one from real data-set and other is sample from noise at each iteration.Stochastic gradient ascent is applied to update discriminator's parameters in a direction to maximize the loss function in solution space and stochastic gradient descent is applied to update generator's parameters to minimize $log(1 - D(G(z)))$ in solution space.There should be some equilibrium point of this game, otherwise this game runs forever. Equilibrium point is achieved when *D(X)* becomes 0.5 or when generator's data distribution pz becomes same as prior data distribution $pdata$.

In experiment, DCGAN(Deep Convolutional Generative Adversarial Network) is chosen.DCGAN[6] is a popular variant of GAN which gives good results in image generation.It composes of mainly convolutional layers without any fully connected layer and max pooling.It contains convolutional strides for down sampling and up sampling.Generator network is a convolutional neural network which contains input layer, four hidden layers which do inverse of convolution operation and output layer.ReLU is used as an activation function.Discriminator network's architecture is same as a generator but performing opposite operations on input data.Instead of ReLU, LeakyReLU is used as an activation function.Batch normalization is applied in both of the network, which is used to normalize all the features in a same dimension.Output layer contain a sigmoid activation function, which outputs 0 or 1.

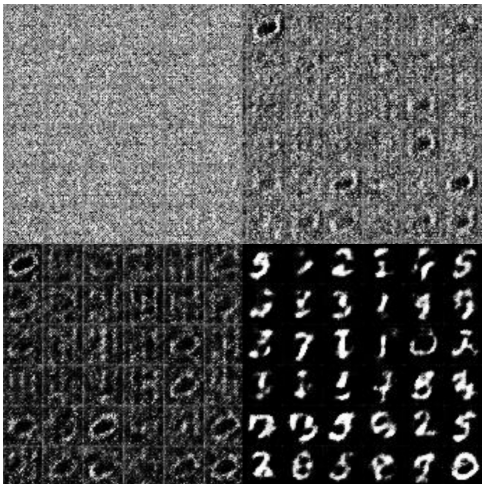

Fig. 5. **Visualization of generated images by DCGAN.Above images are sampled at four different epochs .(i)first epoch (ii) second epoch (iii) fifth epoch (iv) hundred epoch**

BCE(Binary cross entropy) is used as a loss function.Adam optimizer is used to optimize both network.Image quality of generated image increase as number of epoch increase.

Applications:-

—GAN is used as mainly to generate data.But recent advancement in this field leads to various other successful variants of gan which are widely popular for its applications.some of them are shown below.

—In Entertainment such as some animation studios can use DCGAN[6] to generate cartoon characters and background images.

—Fashion industry can use it to input pose to transform it into other poses.

—CycleGan[7] is one of successful variant of gan, which is used to transform images of one domain to another domain, like it can transform pictures between horse and zebra.

—SRGan[8](super resolution gan) is used to generate high resolution images from low resolution image.It shows very impressive results.

—StackGan[9] is another domain transfer model like Cyclegan.It is used to transform input text to generate images that perfectly matches the descriptions

4. COMPARATIVE ANALYSIS

It can be seen from table 1 that DCGAN supports a purely unsupervised learning.Where else, VAE supports both semi supervised and unsupervised learning.Both have convolutional neural network architecture.But in VAE it is encoder-decoder type and in DCGAN it have some constraints like it doesn't have fully connected layers and max pooling layers.Stochastic gradient descent based optimization is used in both the networks for training with adam optimizer.To evaluate performance, log-likelihood is measured in VAE using KL divergence.Mean squared error is also used in both for comparison.

—First criteria to evaluate performance of two generative model is by comparing mean squared error of both of the models.As VAE is taking pixel values of an image as an input, convert it into lower dimensional space and then reverse it to generate original image.The mean squared error of VAE tend to decrease during training and at last it is 34.66 in an experiment.Where else, mean squared error of DCGAN be 36.3 and it is oscillating at every epoch because of the minimax game.So it is difficult to measure performance of DCGAN.Recently various other evaluation metric is discovered such as inception score[13] and FID[14] score to more accurately measure performance of GAN.

—second criteria to compare the two model is by visual inspection of generated samples.As shown in figure, Variational autoencoder tend to produce blurry images compared to the DCGAN.The reason is Decoder output a average value of all generated images or mean value of distribution.L1 loss can be used to reduce blurriness in generated images.

—DCGAN outputs garbage images during some starting epochs as shown in figure 5, because it's loss value is high due to sampling from random noise.The loss measures how well player is doing against the competitor.So after some epochs generator loss increases, but still image quality of generated images increases.

—DCGAN produces high quality images compared to the VAE but limited varieties of samples.It is because equilibrium point sometimes may not be reached.Non convergence is a very difficult problem for GAN.

—GAN sometimes suffers from vanishing gradient problem.It is because in minimax game, Discriminator become more successful compared to generator then cost function will not be properly optimized.So Generator's gradient is very less and it learns nothing due to very short update of parameters. Alternative cost function is proposed for Generator to resolve this problem by author[1]. Instead of $E_{z\sim pz(z)}[\log(1 - D(G(z)))]$,new cost function $\mathbb{E}_{z\sim pz(z)}[-\log(D(G(z)))]$ can be used.

—If in minimax game, Generator become more successful compared to discriminator for minimizing cost function than Discriminator learns nothing due to less gradient update.As a result Generator will be successful to fool the discriminator..An another name of this problem is mode collapsing, in which there is imbalance between modes in generated images.For example for this experiment, if mode collapsing occur than generator will generate some specific digit more compared to other digits.It is still an open research problem to solve.

Table 1. Comparision between two models

Criteria	VAE	DCGAN
Learning type	Semisupervised & unsupervised	Unsupervised
Architecture	Convolutional Autoencoder	Convolutional networks with some constraint
Gradient Update	SGD with update to reconstruction and KL loss	SGD update to both Generator and Discriminator
Optimizer	Adam	Adam
Objective	Inference by matching latent data distribution to original data distribution	Learn structural hierarchy of objects in Generator and Discriminator
Performance Metrics	Log-likelihood and error rate	Accuracy and error rate

5. CONCLUSION

This paper provides a comparative analysis of two popular Generative models on basis of their objective, performance and architecture.Both models have their own pros and cons.Recently computer scientists trying to create a more advance model by combining both of them.One popular model is VAE-GAN[10]. As shown in figure 6, it looks like a variational autoencoder,

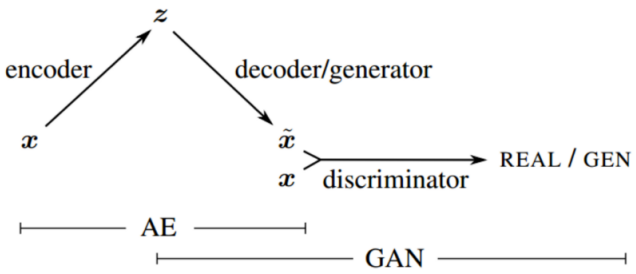

Fig. 6. **Structure of VAE-GAN[10]**

but difference is decoder is replaced with Generator of GAN and loss function is calculated using discriminator.The result is a pretty impressive.Image quality is improved compared to VAE and it outputs more diverse images compared to the GAN.Other models like Adversarial autoencoders[11], SAGAN(Self Attention Generative Adversarial Network)[12] also improves image quality and diversity in generated images compared to VAE and GAN.

6. REFERENCES

[1] Ian Goodfellow, M. Mirza, B. Xu, Y. Benjio. *Generative Adversarial Network*. Department of Computer Science and Research Operationl, University of Montreal(2014).

[2] Ian Goodfellow. *NIPS 2016 Tutorial:Generative Adversarial Networks*, from NIPS conference(2016).

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

[4] Adam Roberts, Jesse E., Douglas E. *Hierarchical Variational Autoencoders for Music*, Google Brain. from NIPS(2017).

[5] David Ha, Douglas E. *A Neural Representation of Sketch Drawings*, from arXiv:1704.03477v4 [cs.NE](2017).

[6] Alec Radford, Luke Metz,Soumith Chintala. *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS*, from ICLR(2016).

[7] Jun-Yan Z.,Taesung P.,Phillip I., Alexei E. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, from arXiv:1703.10593v5 [cs.CV](2018).

[8] Christan L.,Lucas T.,Ference H.Jose C.,Andrew C. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*, arXiv:1609.04802v5 [cs.CV](2017).

[9] Han Z.,Tao X.,Hongsheng L.,Shaoting Z. *StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks*, arXiv:1612.03242v2 [cs.CV](2017).

[10] Hugo L.,Ole W.,Anders L.,Soren S. *Autoencoding beyond pixels using a learned similarity metric*, arXiv:1512.09300v2 [cs.LG](2016).

[11] Alireza M.,Brendan F.,Ian G. *Adversarial Autoencoders*, arXiv:1511.05644v2 [cs.LG](2016).

[12] Dimitris M.,Ian G.,Han Z. *Self-Attention Generative Adversarial Networks*, arXiv:1805.08318v1 [stat.ML](2018).

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

[14] Martin H.,Hubert R.,Thomas U.,Bernhard N. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*, arXiv:1706.08500v6 [cs.LG](2018).