Comparsion Studying Report on GAN and VAE

Jiajing Liao
Computer and Information Science
University of Florida
Gainesville, FL 32603
jiajingliao@ufl.edu

Abstract

Generative model is one of the hottest topic today in Machine Learning. This report studies and comparies the 2 famous generative model: GAN [1] and VAE [2], and give out further analysis about their differences and performances. VAE is a improved autoencoder model which force encoder learns the mapping from images to some latent distribution parameters space, Then, the input to the decoder is some generated random noise with same distribution to the latent parameters. Then decoder learns the mapping from the distribution to images. However, GAN is different. The generator will try to learn the mapping from simple random low-dimension noise to images. Then generator play the minimax game with discriminator in order to improve. The code can be find on GitHub: https://github.com/LiaoJJ/GAN-vs-VAE

1 Introduction

Generative Model is a kind of unsupervised learning techniques. There is no ground-truth labels. The neural network will try to learn the latent distribution of images. Then, generate new images through mapping from latent distribution to images [6].

Generative model help us understand the internal semantics of data. Meanwhile, the cost of training a generative model is usually low since it doesn't need any labels.

This report will compare and study 2 famous generative model. The first one is Variational Auto-Encoder, VAE, and the second one is Generative Adversarial Nets, GAN. Then this report conduct a comparsion between them. Below is a picture I found on the internet which is a good summary for the architecture of GAN and VAE.

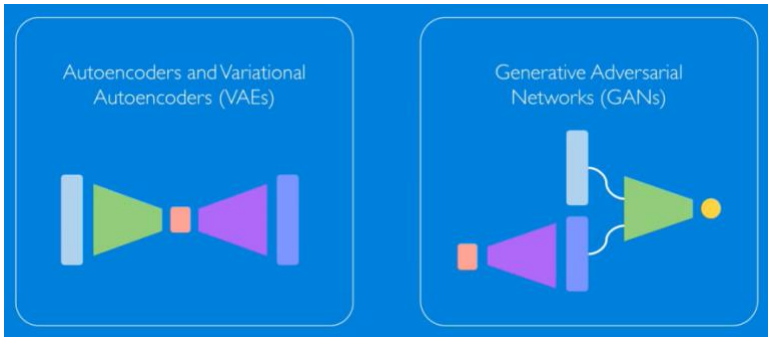

Figure 1: VAE & GAN structures [6]

2 Variational Auto-Encoder

2.1 Autoencoders

the basis of VAE is from Autoencoders. Firstly, input the image and do several times of downsampling, reduce the size of nodes. Then, Compress the input image into a small latent space. After that, decode and upsample the latent node, enlarge the size of nodes. Finally, reconstruct the output in the same size with origimal image.

As below equation, the goal of autoencoders is uaually minimize the difference between the reconstructed images with the original image.

$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Experiments on autoencoders shows that the size of latent space leads to siginificant impact to the quality of reconstructed images. It's more likely a form of compression and the smaller size will be a bottleneck.

2.2 VAE

In order to solve the problem of Autoencoders, Diederik introduce the VAE, which introduce probabilistic perspective into the model.

Instead of a latent space z, VAE seperate the latent space into 2 seperate latent parameters, μ for mean vector and σ for standard deviation vector. μ and σ is parameters for some prior probability, e.g. Gaussian. Then latent space z will be generated randomly from the distributation of μ and σ. As a result, the duty of encoder will be computing the posterior p(z|x), and the duty of decoder will be computing q(x|z) [6].

2.3 Model Architectural

I build a VAE model with 6 layers based on [9]. The structure of the model is as below picture, the number is the number of nodes in each layer. Each layer mainly consist of 2 parts, Fully Connected layer and ReLU Activation layer.

784 - 400 - 20 - 20 - 400 - 784

| /

20

Figure 2: VAE CNN structures

The goal of the entire model will be minimize the difference of original imagesre with constructed images, which has the same distributation with the original images. The first loss is reconstruction loss and the second loss is regularization loss. The first loss is obtained from the difference between original image and generated image. The second loss is KL-divergence between p(z|x) and p(z) in order to avoid overfitting [6].

Reparametrizing is a trick used in VAE training. In order to backpropagate the gradient to the encoder through latent layer z, VAE use below equation to generate z and finish the backpropagation. Hence, z will be randomly generated by Gauss distribution as well as parameters from encoder.

$$z = \mu + \sigma \odot \epsilon, \epsilon \sim N(0, 1)$$

2.4 Experients

2.4.1 Training

I train them 100 epoch, with a batch_size = 64. More detailed training parameters can be found in the code.

During the training, I've tried many different parameters and train them extensively in order to obtain the best hyper-parameters as well as the best result. At the beginning, my model usually doesn't work or doesn't converge. I cahnge my model structure a lot to fix these problems. Finally, I uses a relatively simple architecture in order to speed up the training. It still takes hours to finish the training.

As Figure 3, the loss of VAE reduce faster at the beginning the training. Later, it reduce very slowly.

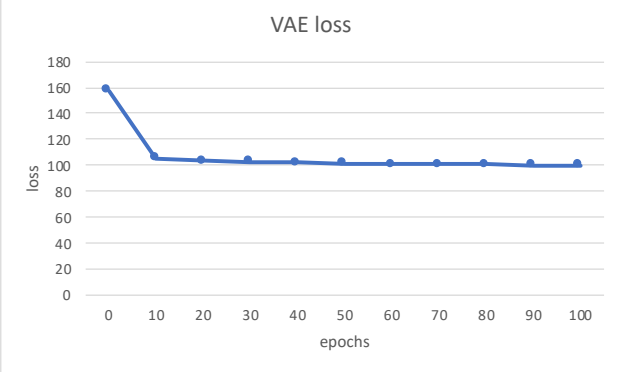

Figure 3: VAE loss

2.4.2 Pertubation Test

In the experiment, we observed a transition from "5" to "3" to "2" through changing the distribution of input random noise gradually.

Figure 4: Pertubation Test Result

2.4.3 Result Analysis

As you can see from the below result images, some of the generated images looks like real numbers. However, some of generated data looks like weired. The images looks smooth. The overall quality is good.

Figure 5: VAE generated images result

3 Generative Adversarial Nets

3.1 Introduction

Unlike the VAE which tries to learn the latent distribution from images, GAN generate images from simple random noise directly. GAN tries to learn a mapping from simple low dimension noise to high dimension images.

GAN is consist of 2 seperate modules, G stands for Generator and D stands for discriminator. Generator maps input simple random noise to generated fake image. Discriminator distinguish input images as read or fake. The goal of the generator is to generate as read as possible images and the goal of the discriminator is to distinguish real and fake images as precise as possible. As the training goes on, Generator and Discriminator will improve together.

This can be model as a minimax game. Equilibrium is a saddle point of the discriminator loss. Generator

minimizes the log-probability of the discriminator being correct.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Figure 6: minimax equation of GAN [1]

3.2 Model Architectural

I build GAN with following model architecture based on [8], the number represents the number of nodes in each layer. Each layer is consist of full connect layer, batchnorm layer and ReLU layer.

Generator is consist of 5 layers: 100 - 256 - 512 - 1024 - 784.

Discrimnator is consist of 5 layers: 784 - 1024 - 512 - 256 - 1.

Figure 7: GAN CNN structures

3.3 Experients

3.3.1 Training

I train them 100 epoch, with a batch_size = 64. More detailed training parameters can be found in the code.

As Figure 8, the loss of discriminator reduce at the beginning, and then start to grow. This makes sense because as the distribution of generated images by generator gets more precise, it's more and more hard for discriminator to distinguish them. Meanwhile, the loss of generator shrinks pretty slow, it keeps reducing fast during the first 40 epochs. Then it reduce slowly after that.

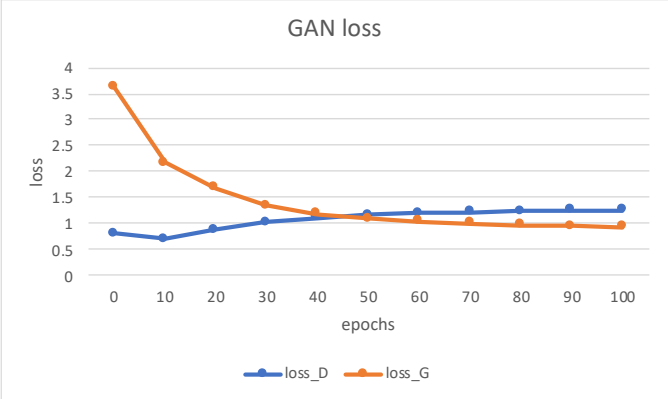

Figure 8: GAN loss

3.3.2 Pertubation Test

In the experiment, we observed a transition from "1" to "8" to "3" through changing the distribution of input random noise gradually.

Figure 9: Pertubation Test Result

3.3.3 Result Analysis

As you can see from the below result images, some of the generated images looks like real numbers. However, some of generated data looks like weired. The image contains some white dot noise. The possible reason is that these white dot noise contribute less to the loss of discriminator.

Figure 10: GAN generated images result

4 Comparsion

VAE is a improved autoencoder model which force encoder learns the mapping from images to some latent distribution parameters space, Then, the input to the decoder is some generated random noise with same distribution to the latent parameters. Then decoder learns the mapping from the distribution to images.

However, GAN is different. The generator will try to learn the mapping from simple random low-dimension noise to images. Then generator play the minimax game with discriminator in order to improve.

1. For the latent space, the latent space of VAE is obtained from encoder, however, the latent space is simple low-dimension random noise.

2. For the ideas, the core idea of VAE is learn latent distribution inside images; however, the core idea of GAN is compete the minimax game.

3. For architecture, The VAE is consist of encoder and decoder, but the GAN is consist of generator and discriminator.

4. From my human-eye point of view, the overall quality of GAN is better than VAE. The images of VAE is more smooth, but the some images looks more weird. Although some images of GAN contains white dot noises, it looks more like numbers.

5. The training of VAE is simpler than GAN, it converge much faster. The Generator of GAN converge pretty slow.

References

[1] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In Advances in neural information processing systems, pp. 2672-2680. 2014.

[2] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[3] Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).

[4] Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen et al. "Pytorch: An imperative style, high-performance deep learning library." In Advances in neural information processing systems, pp. 8026-8037. 2019.

[5] Ketkar, Nikhil. "Introduction to pytorch." In Deep learning with python, pp. 195-208. Apress, Berkeley, CA, 2017.

[6] Amini, Alexander. "Introduction to deep learning." MIT 6 (2019): S191., https://youtu.be/rZufA635dq4

[7] Linder-Norén, E., PyTorch-GAN, 2019, GitHub repository, https://github.com/eriklindernoren/PyTorch-GAN

[8] Lee, A., pytorch-mnist-GAN, 2018, GitHub repository, https://github.com/lyeoni/pytorch-mnist-GAN

[9] pytorch, examples, 2020, GitHub repository, https://github.com/pytorch/examples