Milestone Report of DCGAN and Wasserstein GAN

Lifan Wang
New York University
lw2435@nyu.edu

Kuan Chen
New York University
kc3422@nyu.edu

Guanhua Chen
New York University
gc2229@nyu.edu

1. Introduction

GAN has shown great results in many generative tasks to replicate the real-world rich content such as images, human language, and music. It is inspired by game theory: two models, a generator and a critic, are competing with each other while making each other stronger at the same time. However, it is rather challenging to train a GAN model, as people are facing issues like training instability or failure to converge. In our recent work, we conducted experiments and researches on several types of gan such as DCGAN, WGAN. And we apply these three GANs to cifar10 and Mnist dataset and compare the results. In the next sections, we will detail the methods we use and the final conclusions.

Figure 1. Mnist Dataset

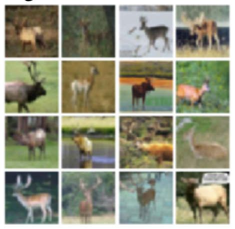

Figure 2. Cifar Dataset

2. Related Work And References

GANs [1] have been known to be unstable to train, often resulting in generators that produce nonsensical outputs.There has been very limited published research in trying to understand and visualize what GANs learn, and the intermediate representations of multi-layer GANs.

The work of Wasserstein GAN[2] introduces a new algorithm named WGAN, an alternative to traditional GAN training. In this new model, it shows that it can improve the stability of learning, getting rid of problems like mode collapse, and provides meaningful learning curves useful for debugging and hyperparameter searches.The work of DCGANs[3] introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning.

From these papers, we can know the concept of GANs is not that hard to understand. But implementing them to produce quality images can be tricky. Our goal is to compare different types of gan and produce better images.

3. Comprehension, Formulation and Goal of Project

Traditional GANs suffer from unstable training progress, such as being hard to achieve Nash equilibrium between generator and discriminator. Even when the training progress runs smoothly, we can still not capture image details because of dimensionality collapsing.To overcome the weaknesses, an advanced structure called Wasserstein GAN[2] implemented a more meaningful loss function, that is, they first replace the old Jensen-Shannon divergence with Wasserstein distance. Then they apply Kantorovich-Rubinstein duality to decrease the model complexity to polynomial time. Finally they clamp the weight of the loss function within a small interval to guarantee K-Lipschitz Continuity.

Traditional GANs uses JS divergence as loss function, where the discriminator tries to propagate the real distribution and inhibit the fake distribution, meanwhile the generator tries to increase the fake distribution.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)}\ {}_{\gamma}[||x - y||]$$

Wasserstein metric shown above is proposed to replace JS divergence because it has a much smoother value space. The WGAN algorithm attempts to train the critic f relatively well before each generator update.

Our goal is to generate some interesting images and acquire a good quality under the evaluation of WGAN, DCGAN and . Then try gradient penalty method to guarantee

K-Lipschitz Continuity.I believe as our project move further in the future, we could have more specific goals.

4. Methods and Algorithms

For method and algorithm, we mainly follow the theory and algorithm of Wasserstein GAN[2]. The comprehension part points to the fact that Earth-Mover(Wasserstein metric) might have much better properties. But its not easy to calculate the Wasserstein metric. We need use K-Lipschitz function to transfer it to the problem:

$$\max_{\omega\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x\sim p(z)}[f_w(g_\theta(z))]$$
$$\leq \sup_{||f||_L\leq K} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x\sim p(z)}[f_w(g_\theta(z))] = K\cdot W(\mathbb{P}_r,\mathbb{P}_\theta)$$

Then it can be proved that this process is principled under the optimality assumption. Hence the loss of discriminator and generator can be calculated.

$$\nabla_\theta W(\mathbb{P}_r,\mathbb{P}_\theta) = -\mathbb{E}_{z\sim p(z)}[\nabla_\theta f(g_\theta(z))]$$

Then we will follow Wasserstein Generative Adversarial Network algorithm[2] to build our model.

Here is our brief algorithm:

For _ in enough iteration

1. we train D for 1 time, using images coming from the Real and Synthesis data. And maximize loss function
2. we train G for 1 time ,using random noise and compare it with Real data, then minimize loss function
3. if loss function == 50%, then end iteration.

5. Technical Depth and Innovation

- We replaced the traditional neural network generator by CNN(ResNet) generator, applied batchnormalization at each level
- We simulated a image scaling stage by using a linear module – a fully connected layer(without batch normalization) , at the entrance/exit of generator/discriminator
- We implemented WGAN-GP by applying gradient penalty into traditional WGANs
- We analyzed the training speed, performance of DCGAN, WGAN, WGAN-GP, and compared their loss function
- We applied WGAN-GP to various dataset of mnist, cifar10, customized dataset.

6. Architecture and Design

The architecture of our project is $"deconv" \Rightarrow "conv"$, where generator is deconv and discriminator is conv.

Here is our design of generator:

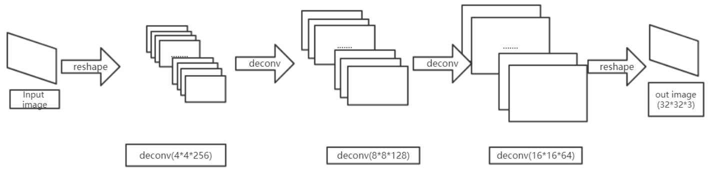

Figure 3. Generator Design

generator: deconv [batch,1024] ⇒ [batch,4096] ⇒ [batch,4,4,256] ⇒ batch,8,8,128] ⇒ [batch,16,16,64] ⇒ [batch,16,16,32] ⇒ [batch,32,32,3]

It reshapes the input image to a 4096 length array. Then after three times deconv, it becomes a 16*16*32 layer. Finally the layer is reshaped to a 32*32*3 output image.

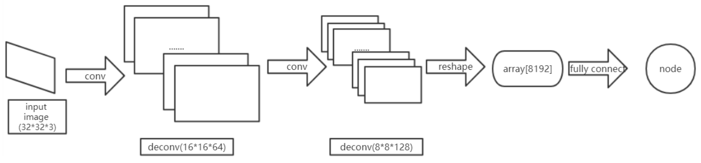

Figure 4. Discriminator Design

Discriminator: conv [batch,32,32,3] ⇒ [batch,16,16,64] ⇒ [batch,8,8,128] ⇒ [batch,8192] ⇒ [batch,1]

After three times conv, it becomes a 8*8*128 layer. Then the layer is reshaped to a 8192 array then fully connected to a tag node.

Three types of Loss Functions : Relax-WGAN loss (EM distance+gradient penalty), WGAN loss(EM distance) , DCGAN loss (JS divergence)

7. Preliminary Results

7.1. Github Repo and Training Abstract

`https://github.com/StarryBar/image_synthesis-WGAN-`

We train DCGAN and WGAN on Mnist, cifar10 and a extra dataset and compare the result and loss of them. The codes and jupyter notebook can be found in the Github repo. Here is some parameters of our training: learning rate = 0.001 β_1 = 0.5

dataset	picture size	traing detail	learning model
mnist	28*28*1	30min on P10, 3000 iterations.	DCGAN
cifar10	32*32*3	3h on P40, 66000 iterations.	WGAN-GP
faces	96*96*3	1.5h on P40, 3500 iterations.	WGAN-GP

7.2. DCGAN Result

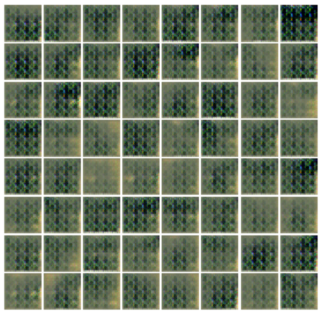

Figure 5. DCGAN on cifar10-deer after 500 iteration

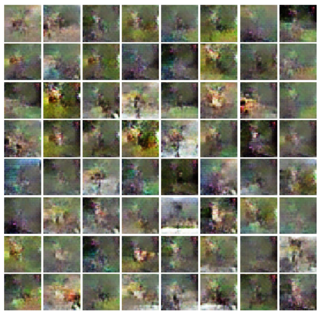

Figure 6. DCGAN on cifar10-deer after 2500 iteration

Due to the time limit, Our DCGAN is not very well tested. Although it can work, its results are not as good as we expect. Following result of WGAN is much better.

7.3. WGAN Result

MNIST dataset:

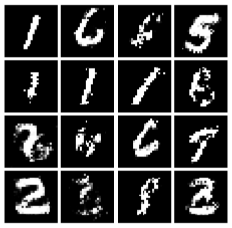

Figure 7. WGAN on MNIST after 2500 iteration

Cifar10-deer dataset:

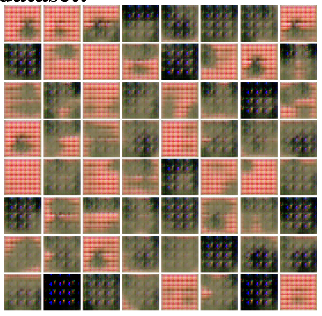

Figure 8. WGAN on cifar10-deer after 1000 iteration

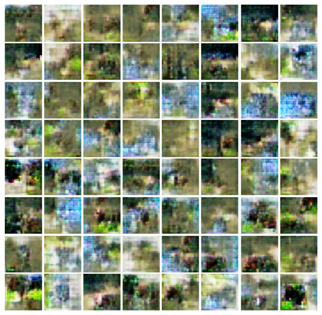

Figure 9. WGAN on cifar10-deer after 4000 iteration

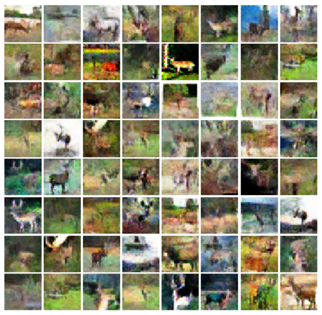

Figure 10. WGAN on cifar10-deer after 60000 iteration

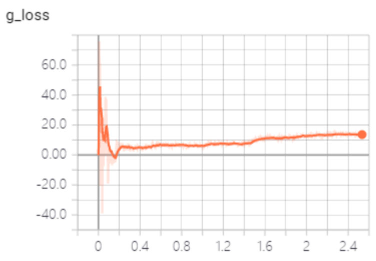

Figure 11. Generator Loss of WGAN for cifar10 dataset

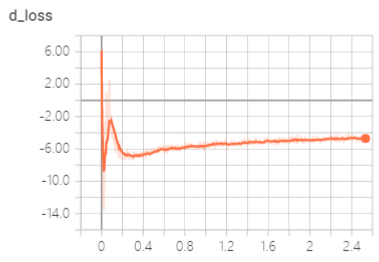

Figure 12. Discriminator Loss of WGAN for cifar10 dataset

From the loss figures, we can see that both generator and discriminator are converged to a small range. And the result pictures seem good. After long enough iteration, we can figure that the result is deer easily.

Comic faces: We also apply the WGAN to a Comic faces dataset. It's a lot of fun and interesting.

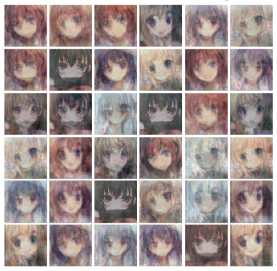

Figure 13. WGAN on COMIC FACE dataset after 1000 iteration

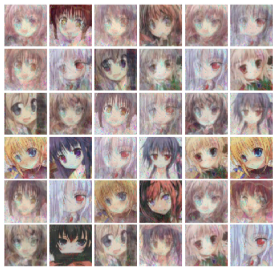

Figure 14. WGAN on COMIC FACE dataset after 3500 iteration

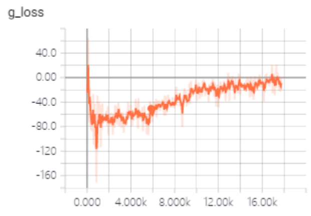

Figure 15. Generator Loss of WGAN for comic faces dataset

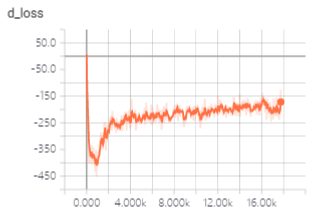

Figure 16. Discriminator Loss of WGAN for comic faces dataset

7.4. Discussion

1. Normally WGAN only need to use fully connected layers for generator, but here we use complicated structure in DCGAN for the generator to achieve better results.

2. Cifar10 dataset is not good for image generation. Firstly, cifar10 dataset has ten labels, each label stands for a different type of image. We cannot put all of the types into training stage. So we choose only one type for training. Secondly, Even if we choose only one label for training, the background for the image varies a lot. So we could not achieve similar performance as mnist dataset whose background is only blackboard.

3. For the comic faces data set, there are 50,000 images, but we only choose 128 images so as to train in a fast way. Unluckily, we find the relevant results to be of low variance.

4. DCGAN requires a strongly restricted architecture for both genertor and the discriminator. Our DCGAN is not so powerful enough so that after some time of training, the model collapse and the loss never changes again.

5. There are two ways to rescale the 96*96 image to feed it into the 32*32 entrance of the generator. One is use OPENCV resize function. Another way is to add another neuron network into the generator. We apply the second method and find that we cannot add batch normalization after the fully connected layer, for the rescaling function is only a linear function but batch normalization destroys the linear model.

8. Reference

[1]Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Networks. arXiv:1406.2661, 2014.
[2] Martin Arjovsky and Soumith Chintala and Lon Bottou. Wasserstein GAN. arXiv:1701.07875, 2017.
[3]Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional

Generative Adversarial Networks. arXiv:1511.06434, 2016.
[4] Xin Guo, Johnny Hong, Tianyi Lin, Nan Yang. Relaxed Wasserstein with Applications to GANs. arXiv:1705.07164, 2017.
[5]LeCun, Yann and Cortes, Corinna, MNIST handwritten digit database, http://yann.lecun.com/exdb/mnist/, 2010.
[6]Alex Krizhevsky and Vinod Nair and Geoffrey Hinton, CIFAR-10 dataset, http://www.cs.toronto.edu/~kriz/cifar.html
[7] Comic Faces dataset, https://cloud.tencent.com/developer/article/1055514.