

Individual Final Report

Chen Chen

INTRODUCTION

Generative Adversarial Network(GAN) is a hot topic in machine learning in recent years, it was invented by Ian Goodfellow and his colleagues in 2014. It has many real life applications in Fashion, Art, Advertising, Science, Video games (picture generation, image conversion, picture repair, video prediction, etc). Image generation is one of the amazing practices of GAN. In this project, a generator that produce various types of anime faces will be developed using Deep convolutional GAN, then the performance of the generator is optimized by Wasserstein GAN (WGAN) and the improved WGAN, which is the WGAN with Gradient Penalty (WGAN-GP).

THE PART I CONTRIBUTE IN THE PROJECT:

- ✓ Topic and data search
- ✓ WGAN-GP model construction

WGAN-GP

WGAN-GP is an improved method of WGAN, which also use the Wasserstein distance to measure the divergence between generated image and real image. The difference between WGAN-GP and WGAN is the weight constrain method. In WGAN-GP, gradient penalty is used instead of weight clipping.

Algorithm¹

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ , the number of critic iterations per generator iteration n_{critic} , the batch size m , Adam hyperparameters α, β_1, β_2 .

Require: initial critic parameters w_0 , initial generator parameters θ_0 .

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_{\theta}(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m -D_w(G_{\theta}(\mathbf{z}^{(i)})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

As mentioned in WGAN, 1-Lipschitz is used to control the discriminator's weight. And 1-Lipschitz function should be everywhere continuously differentiable. So, if the discriminator is 1-Lipschitz, it should have gradients with norm at most 1 everywhere

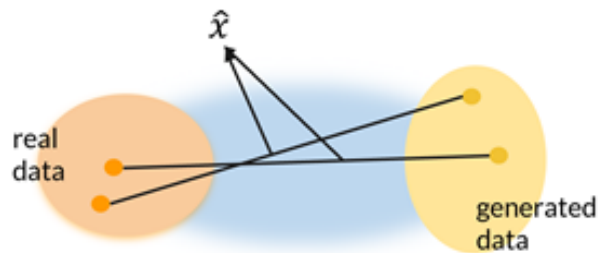
¹ <https://arxiv.org/pdf/1701.07875.pdf>

under the generated image and real image. This is approved by the WGAN-GP author in the paper².

Gradient penalty:

$$\lambda(\|\nabla_{\hat{x}} D_w(\hat{x})\|_2 - 1)^2$$

So points interpolated between the real and generated data should have a gradient norm of 1 for the discriminator. The model will be penalized if the gradient norm moves away from its target norm value 1.



Model Setup

The model structure is almost the same as DCGAN. But here we remove the Sigmoid activation at the last layer in discriminator. Besides, batch normalization in discriminator is also avoided, because it creates correlations between samples in the same batch, which may impact the effectiveness of the gradient penalty. According to the WGAN-GP author, Adam optimizer is used in this model.

Generator Structure

Generator	Output size
Input noise vector: Z(100)	
Linear(100) + BN + ReLu	(512,4,4)
Conv2d_transp(512, 256, (5,5), stride = 2) + BN+ ReLu	(256,8,8)
Conv2d_transp(256, 128, (5,5), stride = 2) + BN+ ReLu	(128,16,16)
Conv2d_transp(128, 64, (5,5), stride = 2) + BN+ ReLu	(64,32,32)
Conv2d_transp(64, 3, (5,5), stride = 2) + Tanh	(3,64,64)

Discriminator Structure

Discriminator	Output size
Input: Image(64*64*3)	
Conv2d (3, 64, (5,5), stride = 2) + LeakyReLu	(64, 32, 32)
Conv2d (64, 128, (5,5), stride = 2) + LeakyReLu	(128, 16, 16)
Conv2d (128, 256, (5,5), stride = 2) + LeakyReLu	(256, 8, 8)
Conv (256, 512, (5,5), stride = 2) + LeakyReLu	(512, 4, 4)
Conv (512, 1, (4,4))	1

Tips for tuning the model:







² <https://arxiv.org/pdf/1701.07875.pdf>

- Increase lambda(coefficient of gradient penalty)
- Train discriminator more than generator

Model performance

Tip 1:

Batch number: 128 Learning rate: 0.0001 Noise:100 Train D once and G once

Epoch	Lambda = 0.5	Lambda = 5
10		
30		
50		

Results: By increasing lambda, the model converges faster, especially at the early stages.

Tip2:

Batch number: 128 Learning rate: 0.0001 Noise:100 Lambda = 5

Epoch	Train D once	Train D twice



Results: By train Discriminator twice, the model improved again. At epoch 50, faces are generated with mouth on the right hand side, but on the left hand side few have mouth.

Model Comparison

WGAN	WGAN-GP
Batch number: 128 Learning rate: 0.0001 Noise:100 Epoch: 50 Train D twice G once Weight clipping: (-0.01, 0.01)	Batch number: 128 Learning rate: 0.0001 Noise:100 Epoch: 50 Train D twice G once Lambda = 5

Results: Both WGAN and WGAN-GP are using the same parameters. WGAN-GP has a higher yield rate than WGAN, and faces are more clear. So WGAN-GP performance is better than WGAN.

WGAN-GP SUMMARY

Using WGAN-GP, the quality of the generated images has been improved, the difference between good and bad images is not that big. However, there are still some faces not generated successfully. This is due to the gradient penalty is still not a strict 1-Lipschitz function, but it works better than the weight clipping method.

Less than 20% is from or copied from the internet.

REFERENCES

Hongyi Lee,

GAN 2018 https://www.youtube.com/watch?v=DQNNMiAP5lw&list=PLJV_el3uVTsMq6JEFPW35BCiOQTsoqwNw

DCGAN <https://github.com/carpedm20/DCGAN-tensorflow>

PyTorch-GAN <https://github.com/eriklindernoren/PyTorch-GAN>

GANHACKS <https://github.com/soumith/ganhacks>

Wasserstein GAN <https://arxiv.org/abs/1701.07875>

Improved Training of Wasserstein GANs <https://arxiv.org/abs/1704.00028>

GAN — Wasserstein GAN & WGAN-GP https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

From GAN to WGAN <https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html>

Machine Learning Mastery <https://machinelearningmastery.com/category/generative-adversarial-networks/>

WGAN-GP <https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/dualgan/dualgan.py>