SPECTRAL NORMALIZATION FOR GENERATIVE ADVERSARIAL NETWORKS

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida

ICLR 2018

Presenter: Yao Zhang

Introduction of paper

My project and experiment

My contribution

Results

Demo

Introduction

GAN is a framework to produce a model distribution that mimics a given target distribution, and it consists of a generator(G) that produces the model distribution and a discriminator(D) that distinguishes the model distribution from the target.

Introduction

GAN is a framework to produce a model distribution that mimics a given target distribution, and it consists of a generator(G) that produces the model distribution and a discriminator(D) that distinguishes the model distribution from the target.

D and G play the following two-player minimax game with value function V (G, D):

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

Introduction

D and G play the following two-player minimax game with value function V (G, D):

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

D maximize: $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$

G minimize: $\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$

Introduction

The machine learning community has pointed out that if discriminator is so good during training, the gradient of generator would vanish.

And some work has done to control the training of discriminator. One way is to control the Lipschitz constant of the discriminator function which is proved to be effective.

for any x, y $$||D(x) - D(y)|| \leq K||x - y||$$

D is Lipschitz continuous and K is Lipschitz constant of D

Introduction

Spectral normalization controls the Lipschitz constant of the discriminator by constraining the spectral norm of each layer.

Spectral normalization normalizes the spectral norm of the weight matrix W:

$$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W).$$

If we normalize weight matrix W of each layer, the function of D is Lipschitz continuous and K = 1.

Introduction

Spectral normalization controls the Lipschitz constant of the discriminator by constraining the spectral norm of each layer.

Spectral normalization normalizes the spectral norm of the weight matrix W:

$$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W).$$

If we normalize weight matrix W of each layer, the function of D is Lipschitz continuous and K = 1.

By definition, $\sigma(W)$ is equivalent to the largest singular value of W.

Introduction

Spectral normalization controls the Lipschitz constant of the discriminator by constraining the spectral norm of each layer.

Spectral normalization normalizes the spectral norm of the weight matrix W:

$$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W).$$

If we normalize weight matrix W of each layer, the function of D is Lipschitz continuous and K = 1.

By definition, $\sigma(W)$ is equivalent to the largest singular value of W.

Use power iteration to estimate $\sigma(W)$ instead of using SVD.

My project and experiment

I run my code on colab.

My project and experiment

I run my code on colab.

I run a standard DCGAN for CIFAR-10 which is well-designed and it uses batch normalization.

My project and experiment

I run my code on colab.

I run a standard DCGAN for CIFAR-10 which is well-designed and it uses batch normalization.

I revise the DCGAN to use spectral normalization which is my second model. No batch normalization is in this model and only use spectral normalization.

My project and experiment

I run my code on colab.

I run a standard DCGAN for CIFAR-10 which is well-designed and it uses batch normalization.

I revise the DCGAN to use spectral normalization which is my second model. No batch normalization is in this model and only use spectral normalization.

My third model is a DCGAN without using batch normalization or spectral normalization.

My project and experiment

I run my code on colab.

I run a standard DCGAN for CIFAR-10 which is well-designed and it uses batch normalization.

I revise the DCGAN to use spectral normalization which is my second model. No batch normalization is in this model and only use spectral normalization.

My third model is a DCGAN without using batch normalization or spectral normalization.

I use FID score as performance metric.

My project and experiment

I run my code on colab.

I run a standard DCGAN for CIFAR-10 which is well-designed and it uses batch normalization.

I revise the DCGAN to use spectral normalization which is my second model. No batch normalization is in this model and only use spectral normalization.

My third model is a DCGAN without using batch normalization or spectral normalization.

I use FID score as performance metric.

The training time is about one and a half hours for one model.

My project and experiment

Generator of DCGAN

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         819,200
       BatchNorm2d-2            [-1, 512, 4, 4]           1,024
              ReLU-3            [-1, 512, 4, 4]               0
   ConvTranspose2d-4            [-1, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [-1, 256, 8, 8]             512
              ReLU-6            [-1, 256, 8, 8]               0
   ConvTranspose2d-7          [-1, 128, 16, 16]         524,288
       BatchNorm2d-8          [-1, 128, 16, 16]             256
              ReLU-9          [-1, 128, 16, 16]               0
  ConvTranspose2d-10           [-1, 64, 32, 32]         131,072
      BatchNorm2d-11           [-1, 64, 32, 32]             128
             ReLU-12           [-1, 64, 32, 32]               0
  ConvTranspose2d-13            [-1, 3, 64, 64]           3,072
             Tanh-14            [-1, 3, 64, 64]               0
================================================================
Total params: 3,576,704
Trainable params: 3,576,704
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 3.00
Params size (MB): 13.64
Estimated Total Size (MB): 16.64
----------------------------------------------------------------
```

My project and experiment

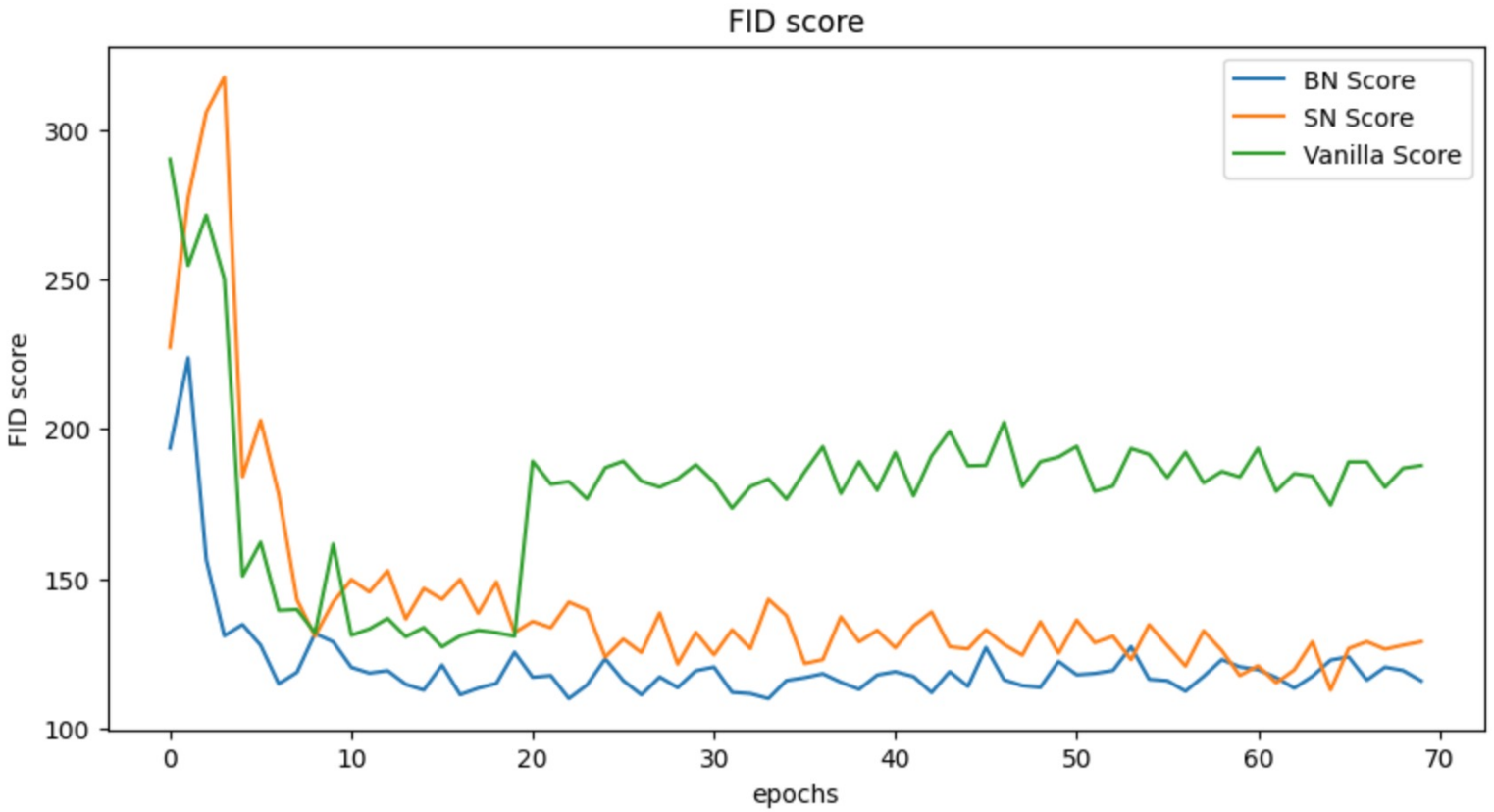
Discriminator of DCGAN

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]           3,072
         LeakyReLU-2          [-1, 64, 32, 32]               0
            Conv2d-3         [-1, 128, 16, 16]         131,072
       BatchNorm2d-4         [-1, 128, 16, 16]             256
         LeakyReLU-5         [-1, 128, 16, 16]               0
            Conv2d-6           [-1, 256, 8, 8]         524,288
       BatchNorm2d-7           [-1, 256, 8, 8]             512
         LeakyReLU-8           [-1, 256, 8, 8]               0
            Conv2d-9           [-1, 512, 4, 4]       2,097,152
      BatchNorm2d-10           [-1, 512, 4, 4]           1,024
        LeakyReLU-11           [-1, 512, 4, 4]               0
           Conv2d-12             [-1, 1, 1, 1]           8,192
          Sigmoid-13             [-1, 1, 1, 1]               0
================================================================
Total params: 2,765,568
Trainable params: 2,765,568
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.31
Params size (MB): 10.55
Estimated Total Size (MB): 12.91
----------------------------------------------------------------
```

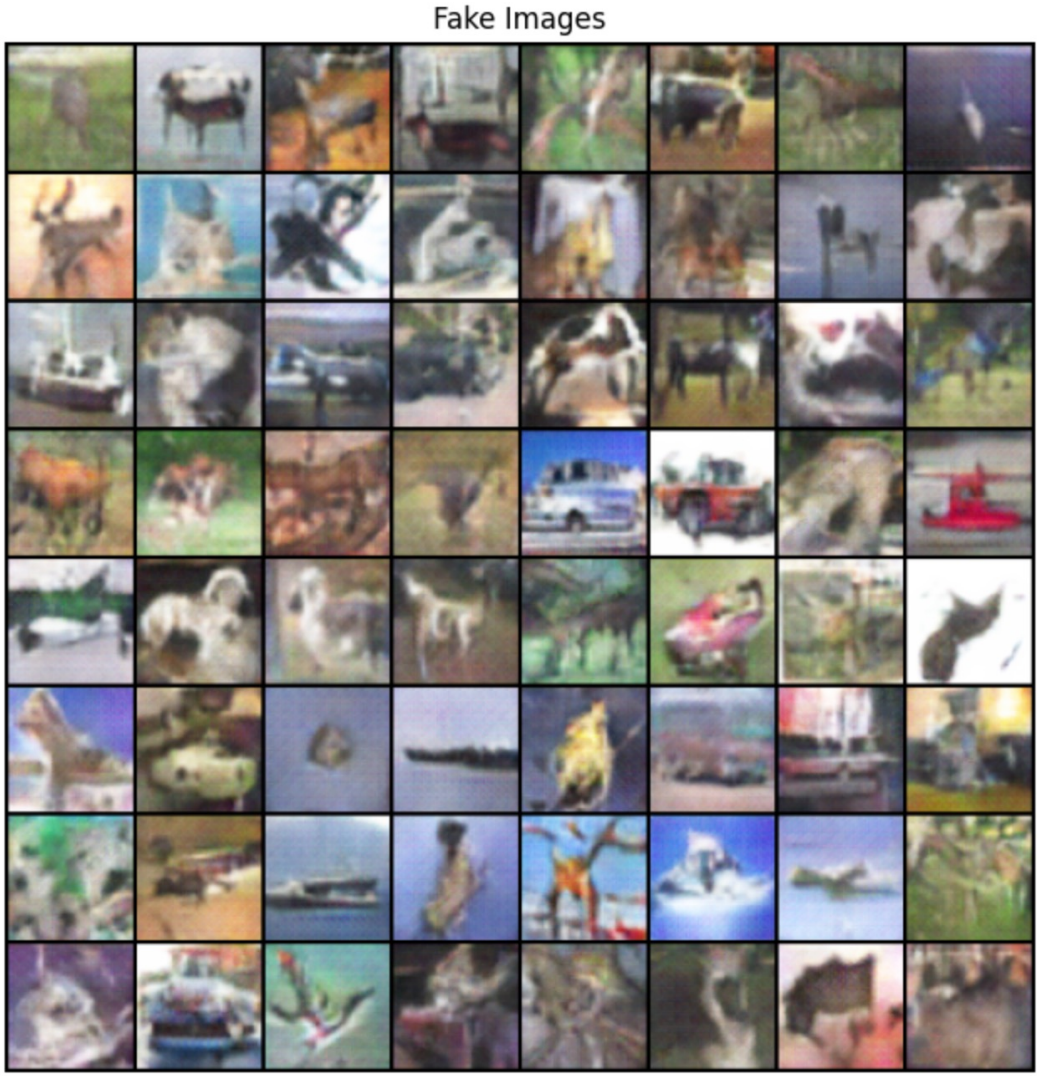
My contribution

The architecture of deep neural networks is different from the architecture used in the paper which can prove the generalization and effectiveness of spectral normalization.

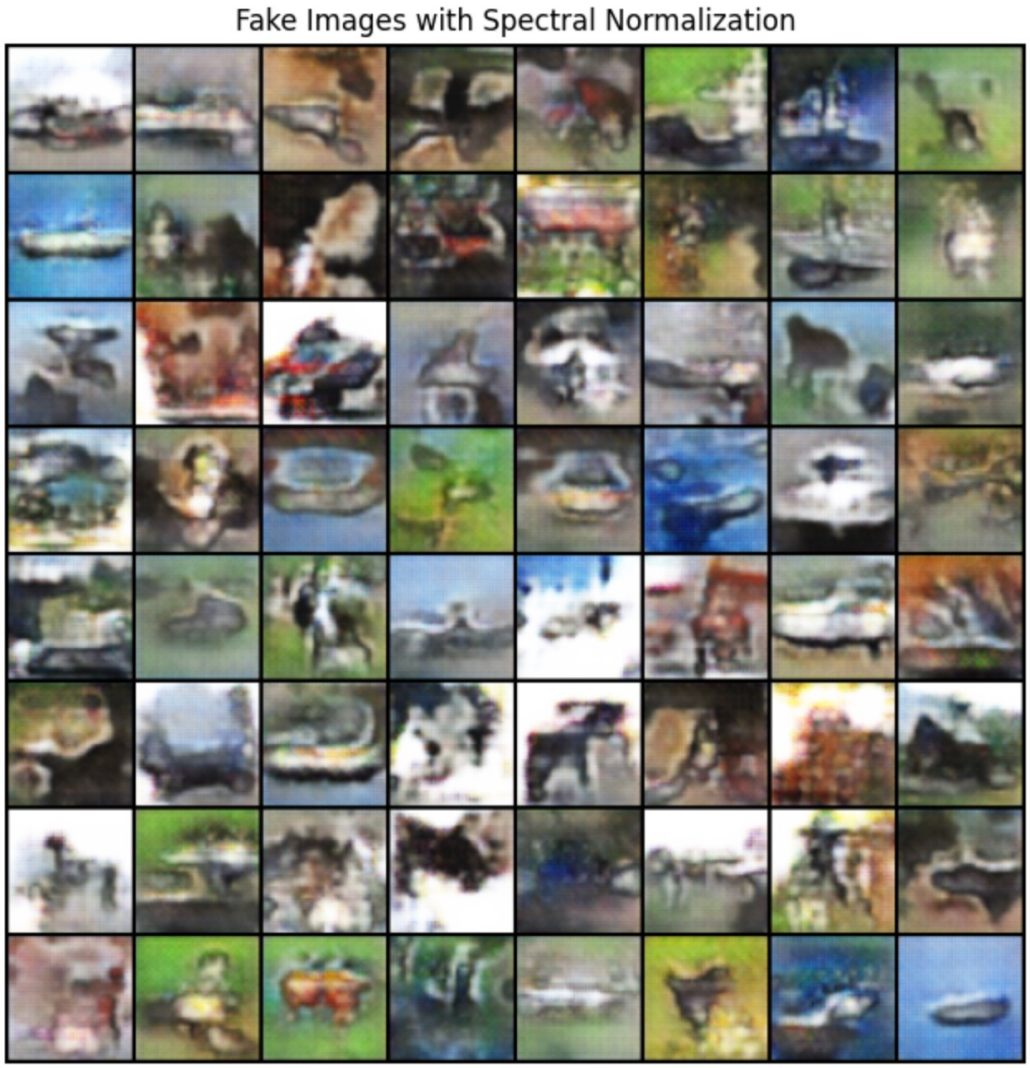
Results


FID score
BN Score
SN Score
Vanilla Score
FID score
epochs
300
250
200
150
100
0
10
20
30
40
50
60
70

Results

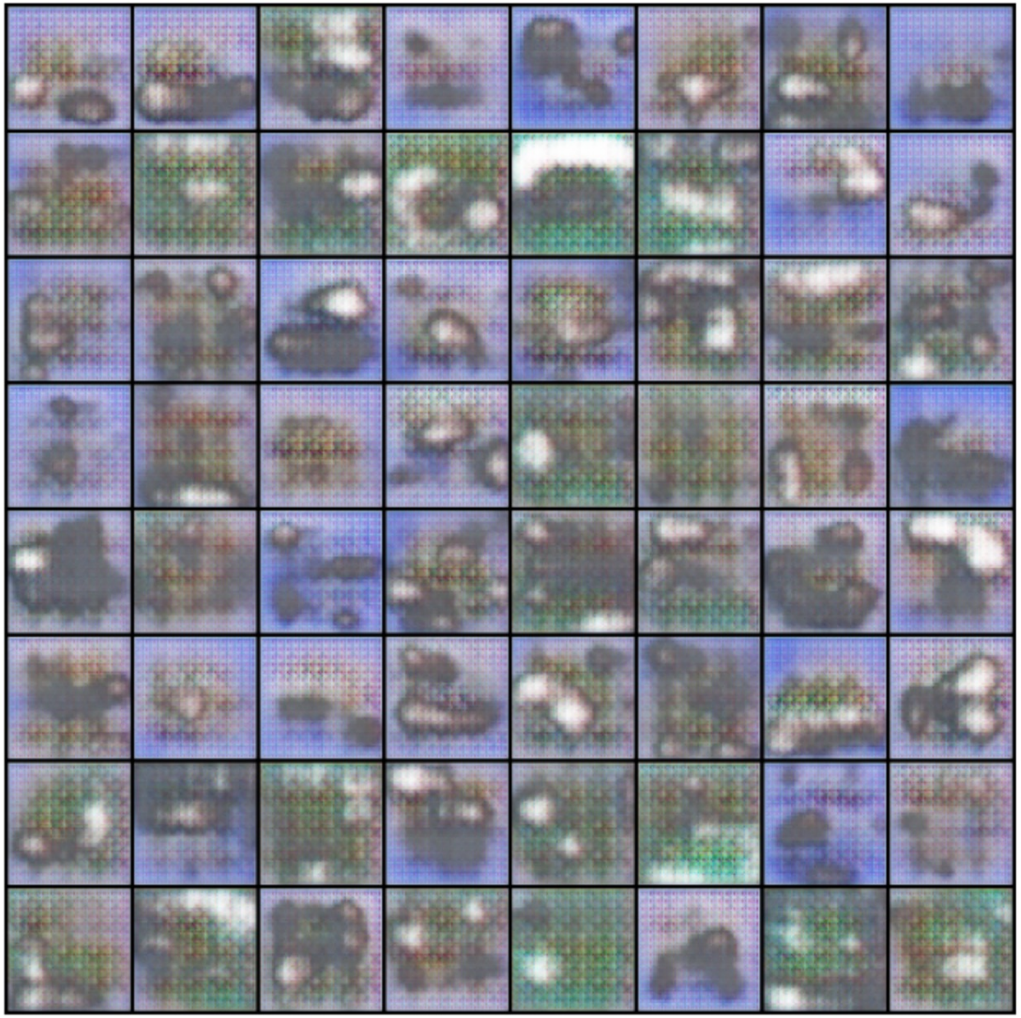


Results



Results

Fake Images without Batch Normalization or Spectral Normalization



Demo

Colab:
https://colab.research.google.com/drive/1_Ktc3XsLH2RpSahY1ly5Sng0EptNQ4VJ?usp=sharing

Github:
https://github.com/yzhan480/CS536Final_Project/