Neural Networks: Final I

刘闵 191240030@smail.nju.edu.cn

Kuang Yaming Honors School, Nanjing University

Basic Settings:

Python 3.8.12 + PyTorch 1.9.1 + CUDA 10.2.89

GPU: GeForce GTX 1080 Ti

1. DCGAN

基础 GAN 模型我采用了 DCGAN。Generator 和 Discriminator 的结构如下：

```
(generator): Generator(
  (l1): Sequential(
    (0): Linear(in_features=100, out_features=8192, bias=False)
    (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (l2_5): Sequential(
    (0): Sequential(
      (0): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (1): Sequential(
      (0): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (2): Sequential(
      (0): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (3): ConvTranspose2d(64, 3, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (4): Tanh()
  )
)

(discriminator): Discriminator(
  (ls): Sequential(
    (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Sequential(
      (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (3): Sequential(
      (0): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (4): Sequential(
      (0): Conv2d(256, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
    (6): Sigmoid()
  )
)
```

Loss function: binary cross entropy

Optimizer: Adam， learning rate=1e-4, (β_1, β_2) = (0.5, 0.999)

Number of Epochs: 20; Number of Steps: 16020

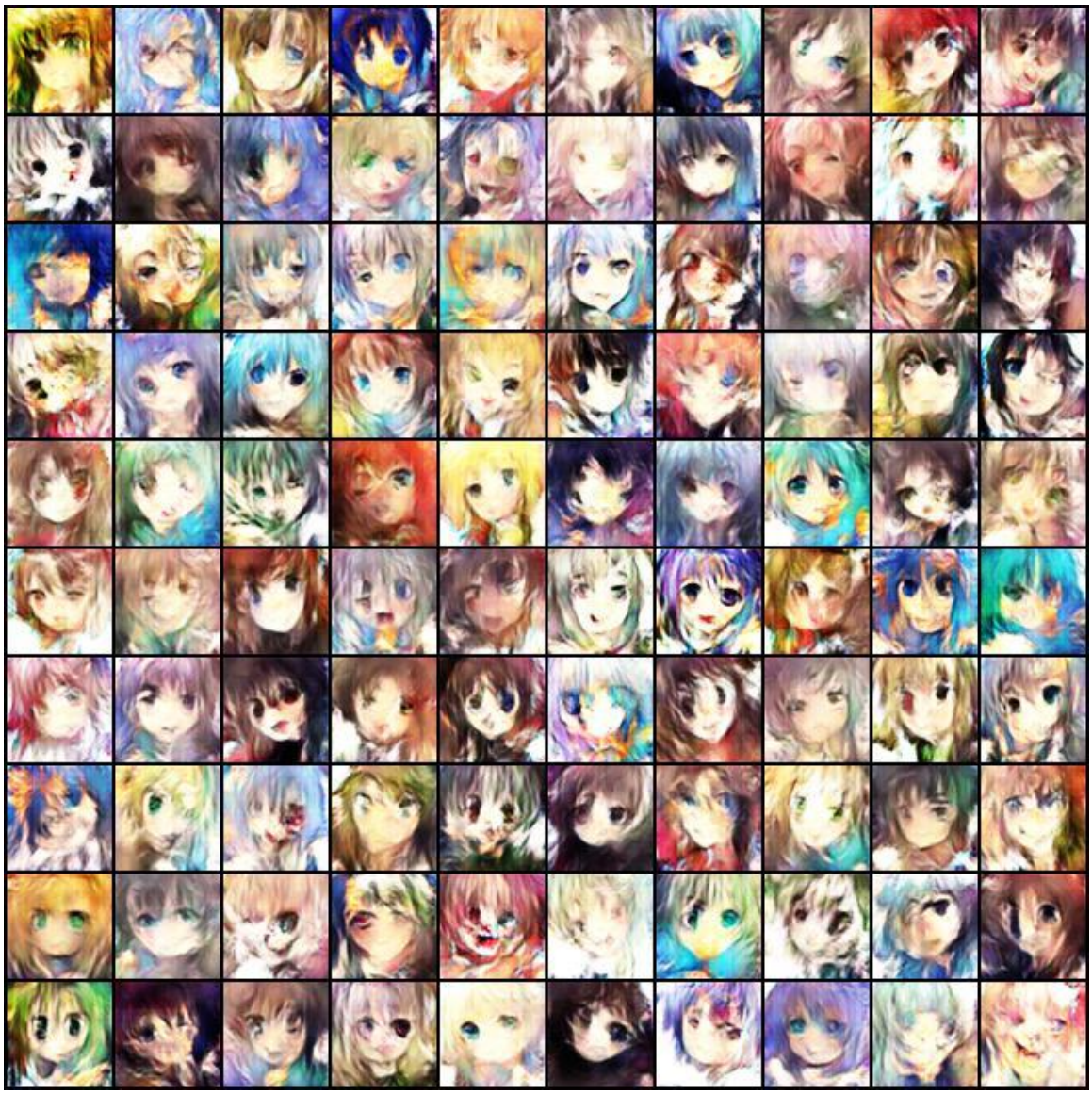
2. WGAN

WGAN 的 Generator 和 Discriminator 结构和 DAGAN 相似，仅在 Discriminator 最后少了一层 sigmoid layer。

Loss function: 由于仅有两个 label (True or False)， 直接用 Discriminator 输出的均值作为 loss function，当真实 label 为 True 时取负，反之取正。

Optimizer: RMSprop，learning rate=1e-4

Number of Epochs: 50; Number of Steps: 40050, generator 每 5 个 step 训练一次



WGAN 比 DCGAN 需要更多的 epoch 来训练，因为它每 5 个 step 才会训练一次 generator。从我的实现效果来看，DCGAN 要好于 WGAN，理论上我继续增加训练次数 WGAN 效果会更好（受限于计算资源，只能训练 50 个 epoch 了）。理论上 WGAN 和 WGAN-GP 不会收敛到纳什均衡，但是会在纳什均衡点附近，最终的训练效果应该和 DCGAN 类似甚至优于 DCGAN。

从训练的结果来看，两种模型都捕捉到了二次元应有的特征，有些图片甚至有异色瞳。多数质量较差的图都已生成了基本的特征，但是五官的比例和细节质量仍然较差。