

Final 3

171098160 王慧敏 商学院金融工程专业

实验采用了 GAN、WGAN 两种模型进行二次元小姐姐头像生成。其中 GAN 又分别采用了全连接网络、卷积神经网络 (DCGAN) 两种方式；WGAN 在卷积神经网络 (DCGAN) 基础上进行修改得到。

一、GAN

GAN 的训练步骤：

对每个 epoch：

对每个 batch：

1、Train Discriminator：

- 固定 generator，训练 discriminator
- 随机生成 batch size 个、latent_dim 维的 input，喂进 generator，得到生成图片
 - 将 batch size 个生成图片、batch size 个真实图片作为 input，喂进 discriminator
 - 真实 label：生成图片的 label=0，真实图片 label=1
 - Loss = BCELoss()，进行梯度下降

2、Train Generator：

- 固定 discriminator，训练 generator
- 随机生成 batch size 个、latent_dim 维的 input，经过 generator、discriminator；
 - 真实 label：生成图片的 label = 0
 - Loss = BCELoss()，进行梯度上升

Optimizer 采用 Adam：

lr = 1e-4

opt_D = torch.optim.Adam(D.parameters(), lr=lr, betas=(0.5, 0.999))

opt_G = torch.optim.Adam(G.parameters(), lr=lr, betas=(0.5, 0.999))

1、全连接网络

网络结构：

Generator:

linear+leakyRelu; 3 * linear+leakyRelu+batchNorm; linear+tanh

```
Generator(  
    (model): Sequential(  
        (0): Linear(in_features=100, out_features=128, bias=True)  
        (1): LeakyReLU(negative_slope=0.2, inplace=True)  
        (2): Linear(in_features=128, out_features=256, bias=True)  
        (3): BatchNorm1d(256, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
        (4): LeakyReLU(negative_slope=0.2, inplace=True)  
        (5): Linear(in_features=256, out_features=512, bias=True)  
        (6): BatchNorm1d(512, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
        (7): LeakyReLU(negative_slope=0.2, inplace=True)  
        (8): Linear(in_features=512, out_features=1024, bias=True)  
        (9): BatchNorm1d(1024, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
        (10): LeakyReLU(negative_slope=0.2, inplace=True)  
        (11): Linear(in_features=1024, out_features=12288, bias=True)  
        (12): Tanh()  
    )  
)
```

Discriminator:

2 * linear+leakyRelu; linear+sigmoid

```
Discriminator(  
    (model): Sequential(  
        (0): Linear(in_features=12288, out_features=512, bias=True)  
        (1): LeakyReLU(negative_slope=0.2, inplace=True)  
        (2): Linear(in_features=512, out_features=256, bias=True)  
        (3): LeakyReLU(negative_slope=0.2, inplace=True)  
        (4): Linear(in_features=256, out_features=1, bias=True)  
        (5): Sigmoid()  
    )  
)
```

训练结果

每 10 个 epoch 查看一下训练结果

- 在训练 10 个 epoch 的时候，可以看出主要学习到了粗略的脸型，出现了眼睛、头发，有些图片中还出现了隐约的嘴的特征，但是除了颜色不同外，脸型、头发、眼睛等特征整体都非常接近；
- 在 50 个 epoch 的时候，与 10 个 epoch 的训练结果差异不是很大；
- 在 100 个 epoch 的时候，出现了一些效果不错的图片，出现了不同颜色的眼睛，发色差异更加明显；



(不同颜色的眼睛)

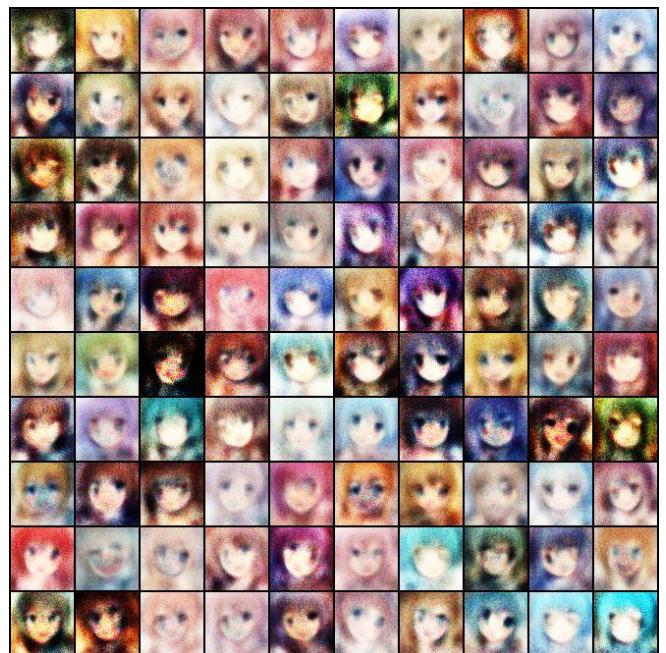
- 在 150 个 epoch 的时候，面部细节更多一些，与 100 个 epoch 改善不大

总体而言，线性模型效果一般，图片整体清晰度很低，发型看不清楚，相似度较高；而且嘴巴这一特征学的不好，很多图片中没有嘴巴

10 个 epoch



50 个 epoch



100 个 epoch



150 个 epoch



2、卷积神经网络（DCGAN）

网络结构特点：

generator、discriminator 采用卷积神经网络

- 在 generator 的网络中，使用反卷积层来代替池化
- 除了 generator 的输出层和 discriminator 的输入层，在网络其它层上都使用了 Batch Normalization，保证稳定学习，有助于处理初始化不良导致的训练问题
- 去除全连接层，直接使用卷积层连接 generator 和 discriminator 的输入层以及输出层
- 在 generator 的输出层使用 Tanh 激活函数，而在其它层使用 LeakyReLU；在 discriminator 上使用 LeakyReLU，输出层使用 Sigmoid 激活函数

Generator:

linear+batchNorm+relu; 3*convTranspose+batchNorm+relu; convTranspose+Tanh

```
Generator(  
    (11): 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()  
    )  
    (12_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:

Conv+leakyRelu; 3*conv+batchNorm+leakyRelu; conv+sigmoid

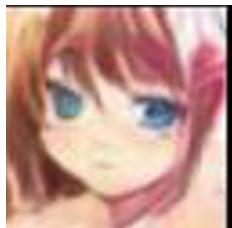
```
Discriminator(  
    (ls): Sequential(  
        (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))  
        (1): LeakyReLU(negative_slope=0.2)  
        (2): Sequential(  
            (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))  
            (1): BatchNorm2d(128, 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()  
    )  
)
```

训练结果：

每 10 个 epoch 查看一下训练结果

- 可以看出，训练效果比线性模型明显改善，在 10 个 epoch 后，已经能够学习到头发、眼睛、脸型这些特征，图片清晰度明显提升，但是很多生成图片中出现了明显的从左上到右下的条纹；
- 在 50 个 epoch 后，发型、眼睛颜色的多样性明显增多，很多图片中都出现了嘴巴的特征，10 个 epoch 中出现的条纹也消失了，但是大部分图片面部表情比较狰狞，或是脸型有一些扭曲
- 在 100 个 epoch 后，生成的效果进一步改善，扭曲的表情数量减少了，生成了许多不错的图片，人物也有了不同的表情、眼睛形状、颜色、脸型等……但有些图片仍有些扭曲，还有些双眼有不同的颜色

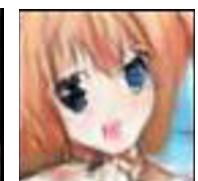
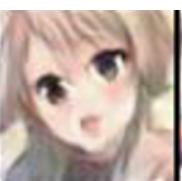
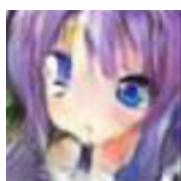
正例：



反例：



- 150 个 epoch，在 100 个 epoch 的基础上稍有改善



总体而言，同样采用传统 GAN 的训练方法，卷积网络模型提取特征能力明显优于线性模型，在生成图片这一任务上，学习效果明显优于线性模型。DCGAN 生成效果不错，生成的图片有不同的脸型、发色、表情、眼睛、嘴形，质量较高

10 个 epoch



50 个 epoch



100 个 epoch



150 个 epoch



二、WGAN

WGAN 与 GAN 的区别在于

- discriminator 最后一层去掉 sigmoid (原始 GAN 的判别器做的是真假二分类任务，所以最后一层是 sigmoid；WGAN 的判别器做的是近似拟合 Wasserstein 距离，属于回归任务，所以要把最后一层的 sigmoid 拿掉)
- 生成器和判别器在算 loss 的时候不取 log
- 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数 c (用来满足 lipschitz 连续性条件)
- 不要用基于动量的优化算法 (例如 momentum 和 Adam)，而多采用 RMSProp、SGD 优化方法

WGAN 的训练步骤：

对每个 epoch:

对每个 batch:

1、Train Discriminator:

固定 generator, 训练 discriminator

- 随机生成 batch size 个、latent_dim 维的 input, 喂进 generator, 得到生成的图片；
- 将 batch size 个生成图片、batch size 个真实图片作为 input, 喂进 discriminator；
- $\text{Loss} = -\text{mean}(\text{discriminator}(\text{真实图片})) + \text{mean}(\text{discriminator}(\text{生成图片}))$
进行梯度下降
- Discriminator 参数更新后，需要 clip weights，保证各个参数的绝对值截断到不超过一个固定常数 clip_value

2、Train Generator:

固定 discriminator, 训练 generator

- 随机生成 batch size 个、latent_dim 维的 input, 经过 generator、discriminator；
- 真实 label: 生成图片的 label = 0
- $\text{loss_G} = \text{torch.mean}(\text{discriminator}(\text{gen_imgs}))$, 进行梯度上升

本文采用 clip_value=0.01

Optimizer 采用 RMSprop:

lr = 5e-5

optimizer_G = torch.optim.RMSprop(generator.parameters(), lr=lr)

optimizer_D = torch.optim.RMSprop(discriminator.parameters(), lr=lr)

网络结构：

Generator:

linear+batchNorm+relu; 3*convTranspose+batchNorm+relu; convTranspose+Tanh

```
Generator(  
    (1): 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()  
    )  
    (12_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:

Conv+leakyRelu; 3*conv+batchNorm+leakyRelu; conv

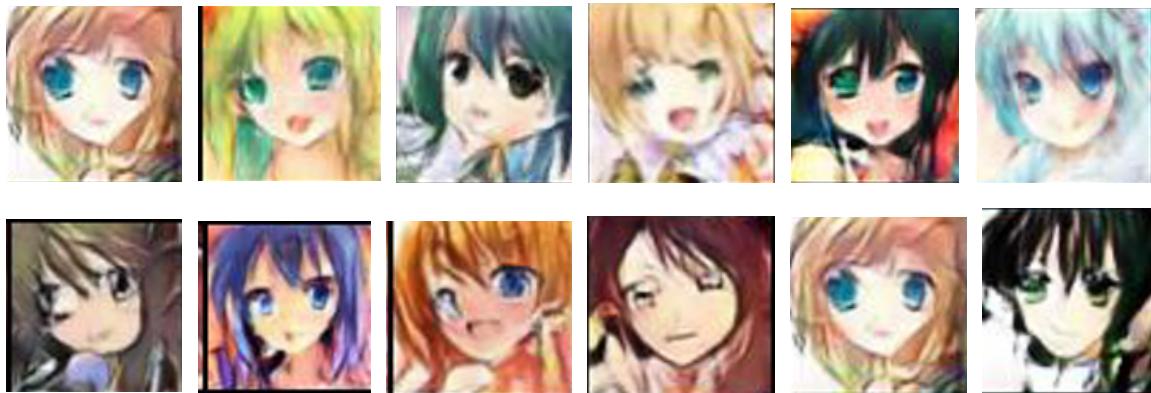
```
Discriminator(  
    (ls): Sequential(  
        (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))  
        (1): LeakyReLU(negative_slope=0.2)  
        (2): Sequential(  
            (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))  
            (1): BatchNorm2d(128, 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))  
    )  
)
```

训练结果：

每 10 个 epoch 查看一下训练结果

- 10 个 epoch 的时候，没有出现 DCGAN 斜向的条纹，但是清晰度不如 DCGAN；已经能够学习到不同颜色的眼睛（DCGAN 10 个 epoch 时基本只有黑色的眼睛），且有了不同的脸型，头发的颜色非常丰富
- 在 50 个 epoch 后，图片较 10 个 epoch 时的清晰度明显提升；相比于 DCGAN 50 个 epoch 时，脸部扭曲的图片更少一些
- 在 100、150 个 epoch 后，生成的效果进一步改善，相比于 DCGAN 同样的 epoch 数，人物脸型、眼睛形状和颜色、表情、嘴巴等的多样性更加丰富，但同样也有一些双眼颜色不同、头发颜色不正常的问题

正例：（150 个 epoch）



反例：（150 个 epoch）

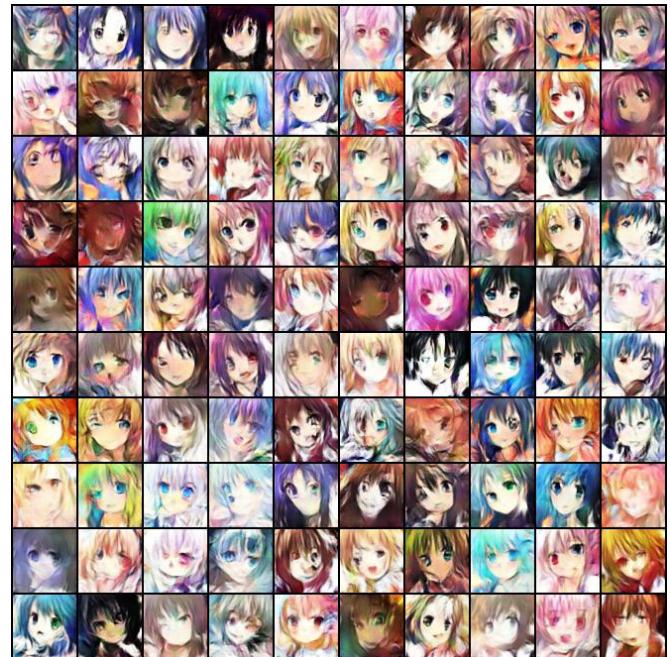


总体而言，WGAN 比相同卷积结构的 GAN 模型（DCGAN）生成效果更好，图片的细节更精致、多样性更丰富，脸型、发色、眼睛、表情、尤其是嘴形，有着更多的样式，生成的质量是三个模型中最高的一个

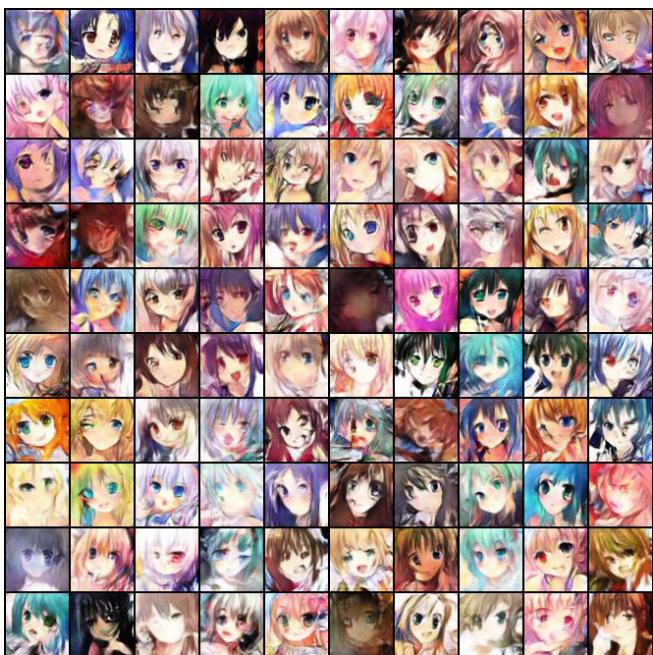
10 个 epoch



50 个 epoch



100 个 epoch



150 个 epoch

