HW3 实验报告：“二次元的你”图像生成

211240074 余今 匡亚明学院脑科学与人工智能方向

1 基础 GAN（GAN_model.py）

1.1 模型理论架构

本次实验使用生成对抗网络（Generative adversarial network, GNN）。GNN 是两个网络的组合：生成网络（Generator）负责生成模拟数据；判别网络（Discriminator）负责判断输入的数据是真实的还是生成的。生成网络要不断优化自己生成的数据让判别网络判断不出来，判别网络也要优化自己让自己判断得更准确。二者关系形成对抗，因此叫对抗网络。

在训练时，Generator 生成假数据，将生成的假数据和真数据都输入 Discriminator 进行判断，根据误差来优化 Discriminator。然后反过来优化 Generator，之后继续优化 Discriminator，循环往复，直到平衡。

1.2 实验中的模型架构

在实验中，我基于理论构建了基础的 GNN 模型，主要包括 MyGenerator 和 MyDiscriminator 两个部分，并根据深度卷积生成对抗网络（Deep Convolution GAN, DCGAN）的理论对模型进行了小改进。

1.2.1 生成器（MyGenerator）

生成器的作用为将随机噪声映射到能拟合真实图片分布的空间，主要由 4 个 gen_block 组成，gen_block 结构如图 1 所示，包括 ConvTranspose2d、BatchNorm2d 和 ReLU。这里使用的 ConvTranspose2d 为转置卷积，可以在开始时进行上采样，BatchNorm2d 可以防止梯度消失和过拟合等。

在 4 个 gen_block 后，增加 Tanh 层，规范输出范围，最终模型结构如图 2 所示。

```python
def gen_block(input_channels, output_channels, kernel_size=4, stride=2, padding=1, bias=False):
    # ConvTranspose2d 每边计算公式: output=(input-1)*stride + kernel_size -2*padding + output_padding
    layer = [
        nn.ConvTranspose2d(input_channels, output_channels, kernel_size=kernel_size, stride=stride, padding=padding,
                           bias=bias),
        nn.BatchNorm2d(output_channels),
        nn.ReLU(True)
    ]
    return nn.Sequential(*layer)
```

图 1 MyGenerator 的 gen_block 结构

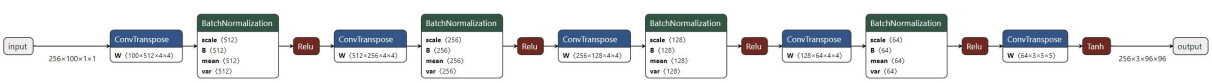

图 2 MyGenerator 结构

1.2.2 判别器（Discriminator）

判别器的作用为分辨真图像与生成器输出的假图像，辅助生成器的训练。在基础的 GAN 模型中，判别器的结构即为简单的二分类模型，MyDiscriminator 主要包括五次卷积操作，在中间三次会加入 BatchNorm2d 来正则化，最后经过 Sigmoid 输出分类结果。由于判别器收敛更快，这里使用 LeakyReLU 来防止梯度消失。最终模型结构如图 3 所示。

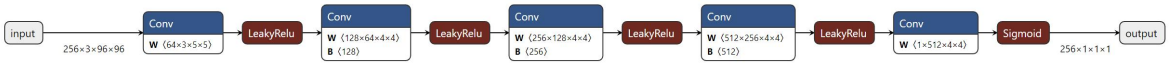

图 3 MyDiscriminator 结构

1.3 训练过程及优化操作

1.3.1 数据预处理

对训练用的图像进行了尺寸和像素值范围的规范，未作其余处理。

```
transform = transforms.Compose([
    transforms.Resize(img_size),
    transforms.CenterCrop(img_size),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

图 4 数据预处理操作

1.3.2 优化器与损失函数

对于生成器和判别器，优化器 optimizer 均采用效果稳定且均衡的 Adam，lr = 2e-4，betas=(0.5, 0.999)。在训练时，真实图像标记为 1，假图像标记为 0，损失函数 loss function 为二分类交叉熵 BCELoss，其对数形式可以实现 GAN 理论的目标函数。

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

```
# 优化器
gen_optimizer = optim.Adam(Generator.parameters(), lr=2e-4, betas=(0.5, 0.999))
dis_optimizer = optim.Adam(Discriminator.parameters(), lr=2e-4, betas=(0.5, 0.999))
# 损失函数用二分类交叉熵
criterion = torch.nn.BCELoss()
```

图 7 优化器与损失函数

1.3.3 训练过程

主要训练过程在 train 函数中，先固定生成器的参数（设置 detach）训练判别器，再冻结判别器的参数训练生成器，一共训练 500 epochs。在综合考虑模型拟合效率和呈现效果后，在难以达到拟合的前提下，每个 epoch 中判别器和生成器的训练次数为 1:1。

1.4 模型训练结果

1.4.1 损失函数结果

在训练了 500 轮后，模型的损失函数变化稳定但还未完全收敛（图 8），考虑到用时问题，提前终止训练，在改进后再采用更多的训练轮次。

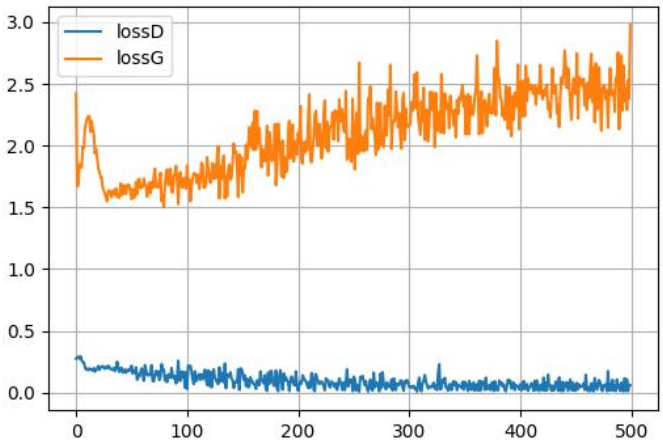

图 8 基础 GAN 模型训练 500 epochs 后的损失变化

1.4.2 生成器效果

在训练了 500 轮后，随机给 16 个初始噪声，生成器的图片生成效果如图 9（200 轮）、图 10（500 轮）所示，200 轮时基本具有人的五官，但是五官比例、大小和位置等效果都不太好，脸部线条也不流畅，个别图片会崩坏得更严重；500 轮时五官位置和画面精细程度会更好一些，但并没有全面优于 200 轮时的，推测是模型还未训练稳定。

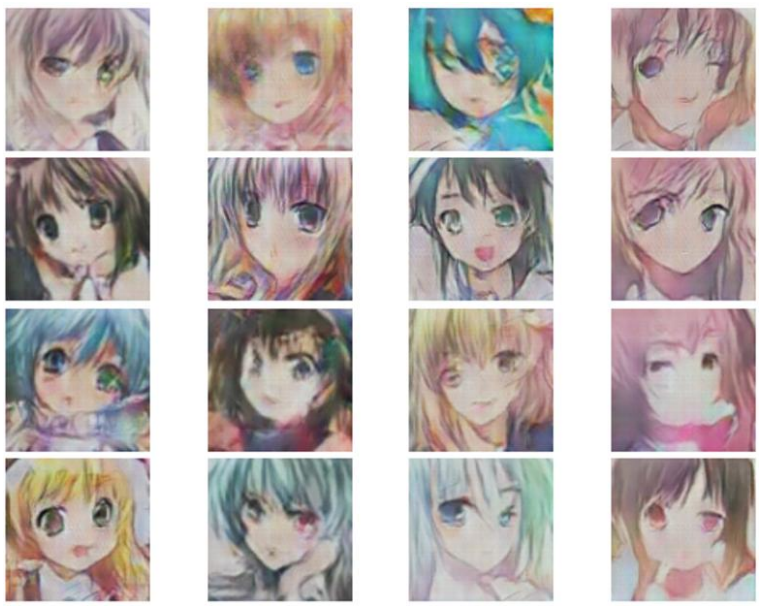

图 9 基础 GAN 模型训练 200 epochs 后的效果

图 10 基础 GAN 训练 500 轮后的结果

2 WGAN-GP 改进模型（WGAN-GP_model.py）

2.1 基础GAN的不足与WGAN-GP的改进

由于普通的 GAN 模型只是单纯衡量真实图片和虚假图片分布领域的重合程度，而在高维空间中，生成器初始化的分布与真实分布重叠部分测度为 0 的概率为 1，这可能导致生成器的梯度消失问题。于是，有人提出了 Wasserstein distance 来衡量生成器分布与真实分布的距离，推动了 WGAN 的发展。

具体的，WGAN 相比 GAN，在判别器中去掉了最后一层的 Sigmoid，对于生成器和判别器的 loss 不取对数，转而计算 Wasserstein distance，同时采用 1-Lipschitz 限制判别器的参数绝对值上界，目标函数更改为。

$$\max_{D\in 1-\text{ Lipschitz}} \left\{E_{x\sim P_{\text{real}}}[D(x)] - E_{x\sim P_G}[D(x)]\right\}$$

此时，基于动量的优化器 Adam 在实验中表现出梯度不稳定的缺点，故更改为 RMSProp。

然而，修改了目标函数后的 WGAN 依旧存在一些问题，如可能会使判别器最终权重集中在极大或极小两个极端，使得参数多样性减少，也会带来梯度消失或梯度爆炸的问题，于是引入梯度惩罚项 gradient penalty 的 WGAN-GP 产生了，使得最终 Discriminator 的目标函数为

$$L = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{x}}}\left[\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2\right]$$

判别器需要最小化自己对假头像的输出，最大化对真头像的输出，同时最小化惩罚项，于是，生成器此时的目标是最大化判别器的输出。

2.2 代码改进

2.2.1 网络结构

在生成器和判别器的网络结构上，我并没有做太大的改变，只删去了判别器最后一层的 Sigmoid 层，保留上一层的输出结果以供计算。除此以外，在定义 MyGenerator 与 MyDiscriminator 时，我增加了权重初始化的模块。

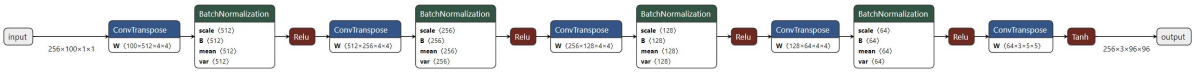

图 11 WGAN-GP 的 MyGenerator 结构

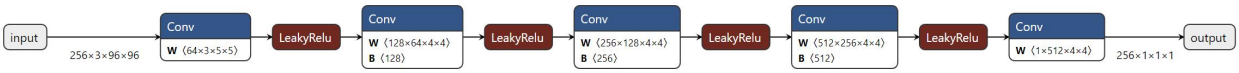

图 12 WGAN-GP 的 MyDiscriminator 结构

2.2.2 优化器 optimizer

由于基于动量的 Adam 在 WGAN-GP 的训练过程中表现得不稳定，所以更改为 RMSProp，参数 lr=2e-4。

```
# 优化: 基于动量的优化算法梯度不稳定, 改成 RMSprop
gen_optimizer = optim.RMSprop(Generator.parameters(), lr=2e-4)
dis_optimizer = optim.RMSprop(Discriminator.parameters(), lr=2e-4)
```

图 13 优化器改进

2.2.3 损失函数 loss function

对于目标函数 L，先计算非惩罚项部分，求得损失非对数形式的均值，再进行反向传播。

```
output = Discriminator(img)
dis_true_loss = output.mean()
dis_true_loss.backward(m_one)

dis_noises = torch.randn(batch_size, noise_size, 1, 1)
dis_noises = dis_noises.to(device)

dis_fake_img = Generator(dis_noises).detach()
dis_fake_output = Discriminator(dis_fake_img)

discriminator_fake_loss = dis_fake_output.mean()
discriminator_fake_loss.backward(p_one)
```

图 14 判别器非乘法项计算过程（生成器类似，取负后反向传播）

对于目标函数的惩罚项部分，使用 calculate_gp 函数进行计算，在真实图像与假图像之间生成插值点并计算梯度，令 λ=10，计算惩罚项并返回，加入目标函数中。

```
def calculate_gp(img, fake_img):
    eta = torch.FloatTensor(batch_size, 1, 1, 1).uniform_(0, 1)
    eta = eta.expand(batch_size, img.size(1), img.size(2), img.size(3)).to(device)
    # 插值得到新图像
    interplt_img = eta * img + ((1 - eta) * fake_img)
    interplt_img = interplt_img.to(device)
    # 需要计算梯度
    interplt_img.requires_grad_(True)
    # 输入判别器得到结果
    prob_interplt_img = Discriminator(interplt_img)
    # 梯度计算
    gradients = torch.autograd.grad(outputs=prob_interplt_img, inputs=interplt_img,
                                    grad_outputs=torch.ones(prob_interplt_img.size()).to(device), create_graph=True,
                                    retain_graph=True)[0]
    # 梯度惩罚项计算
    grad_penalty = ((gradients.norm(2, dim=1) - 1) ** 2).mean() * 10
    return grad_penalty
```

图 15 用于计算梯度惩罚项的 calculate_gp 函数

2.3 训练结果

2.3.1 损失函数结果

在训练了 500 轮后，模型的损失函数变化稳定但还未完全收敛（图 16），中间出现不稳定的情况，但最后效果还算差强人意，考虑到用时问题，提前终止训练。

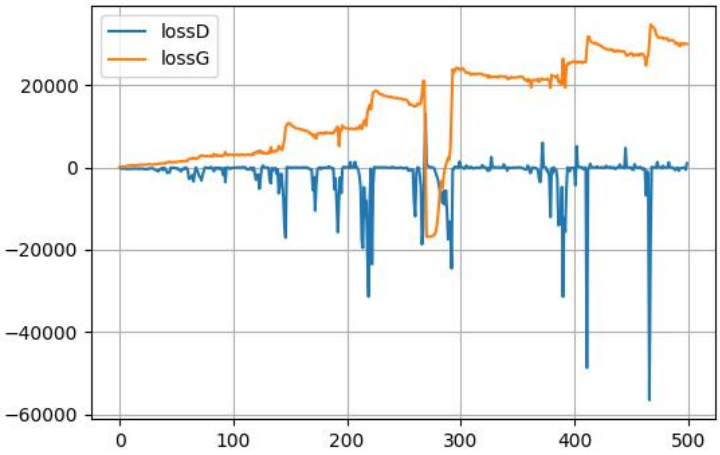

图 16 WGAN-GP 训练 500 轮后的结果

2.3.2 生成器效果

在训练了 500 轮后，随机给 16 个初始噪声，生成器的图片生成效果如图 17（200 轮）、图 18（500 轮）所示，200 轮时基本具有人的五官，但是五官位置、脸部流畅性不够好，偶尔有效果很好的图片，而个别图片会崩坏得更严重；500 轮时五官、头发和画面都更精细，生成图像的平均水平也比 200 轮时好，但个别部分并没有全面优于 200 轮时的，推测是模型还未训练稳定，但总的效果比上文基础 GAN 同期优秀。

图 17 WGAN-GP 模型训练 200 epochs 后的效果

图 18 WGAN-GP 模型训练 500 epochs 后的效果

3 分析与总结

3.1 模型性能

关于基础 GAN 的理论不足与 WGAN-GP 的优势已在 2.1 中作了介绍。

需要补充的是，WGAN-GP 的改进规避了传统 GAN 训练中由于 Discriminator 收敛过快而带来的“Perfect Discriminator”的问题，使得训练者不必为了平衡模型性能与收敛速度而将判别器与生成器的训练轮次比当做超参去调节，提高模型训练的确定性并降低开销。

3.2 收敛效率

WGAN-GP 相比于最初的 GAN，收敛更快。在 WGAN-GP 训练至第 100epoch 时，输出的图像质量与 GAN 第 150 和第 200 轮时的图像质量类似。这可能是因为 Wasserstein Distance 在处理两个几乎无重叠的高维分布之间的“距离”时，比 JS-Divergence 更具有优势。

3.3 图像质量与潜在问题

关于 GAN 与 WGAN-GP 图像的单独分析在 1.4.2 和 2.3.2 中已经给出，下面对这两个模型进行横向对比。

可以看出，在轮次相同时，从画面精细程度来说，WGAN-GP 会优于 GAN，表现为线条更清晰、着色更鲜明。尽管从平均发挥上来说，在 200 轮时，GAN 似乎优于 WGAN-GP，但随着训练轮次的增加，WGAN-GP 则越来越优秀。

从“Mode Collapse”问题的角度分析，在 200 轮时，GAN 表现得尤为明显，有数张图片是类似的，说明 GAN 只能将噪声映射到一个比较小的“图像区间”，多样性能力不足，这种情况在 500 轮时有所改善。WGAN-GP 也存在这种问题，但总体表现比 GAN 轻。

除此以外，由于训练轮次较少和手段限制，无法检测“Mode Dropping”的问题。其次，或许可以用漫画人脸目标检测模型的隐藏层数据来对生成效果作出定量的衡量，我希望在之后进行尝试。