哈爾濱工業大學

模式识别与深度学习实验报告

实验 五

题　　目　生成式对抗网络

学　　院　计算学部

专　　业

学　　号

学　　生

任课教师　左旺孟

哈尔滨工业大学计算机科学与技术学院

2023 年春季

实验五 报告

一、 实验目的

1.基于 Pytorch 实现 GAN、WGAN、WGAN-GP。拟合给定分布（分布由 points.mat 给出）。要求可视化训练过程。

2. 基于给定的 ProGAN 代码和模型，实现隐空间语义方向搜索的代码的 SeFa 部分，完善 genforce/sefa.py 中 TODO 部分。

二、 实验环境

硬件配置（华为云服务器）：

CPU: 8 核 64GB

GPU: 1*P100(16GB)

软件配置：

OS：ubuntu18.04

Packages in Environment：pytorch1.8-cuda10.2-cudnn7

三、 实验内容

3.1 数据集处理

points.mat 点集数据：

调用 mat4py 库读入点集文件，最后调用 DataLoader 实现数据集迭代器的构造。

```
class Points(Dataset):
    def __init__(self):
        self.data = mat4py.loadmat(args.data_dir)['xx']

    def __getitem__(self, idx):
        xy = torch.tensor(np.array(self.data[idx])).to(torch.float32)
        return xy

    def __len__(self):
        return len(self.data)
```

3.2 GAN 神经网络

GAN 训练的过程是一个生成器和判别器博弈的过程。生成器生成假数据，然后将生成的假数据和真数据都输入判别器，判别器要判断出哪些是真的哪些是假的。判别器第一次判

别出来的肯定有很大的误差，然后我们根据误差来优化判别器。现在判别器水平提高了，生成器生成的数据很难再骗过判别器了，所以我们得反过来优化生成器，之后生成器水平提高了，然后反过来继续训练判别器，判别器水平又提高了，再反过来训练生成器，就这样循环往复，直到达到纳什均衡。

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(10, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True),
            nn.Linear(128, 256),
            nn.ReLU(True),
            nn.Linear(256, 2),
        )

    def forward(self, z):
        output = self.net(z)
        return output
```

```
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(2, 64),
            nn.LeakyReLU(),
            nn.Linear(64, 128),
            nn.LeakyReLU(),
            nn.Linear(128, 64),
            nn.LeakyReLU(),
            nn.Linear(64, 1),
            nn.Sigmoid()
        )
    def forward(self, x):
        output = self.net(x).view(-1)
        return output
```

损失函数：

```
# 判别器损失反向传播，越接近0越小，判别效果越好
loss_D = - (torch.log(pred_real) + torch.log(1. - pred_fake)).mean()
# 生成器损失越接近负无穷越小，判别效果越好
loss_G = torch.log(1. - pred_fake).mean()
```

3.3 WGAN 神经网络

原始 GAN 问题的根源可以归结为两点，一是等 价优化的距离衡量（KL 散度、JS 散度）不合理，二是生成器随机初始化后的生成分布很难与真实分布有不可忽略的重叠（上升到高维时）从而导致在判别器优化的很好的时候生成器会有梯度消失的情况发生。

因此，WGAN 利用 EM 距离代替 JS、KL 散度从而解决了生成与真实分布的距离衡量，从而改进了原始 GAN 存在的两类问题。

损失函数如下所示，其中将参数 clip 到[-0.1,0.1]的范围内，args.CLAMP=0.1：

```
if args.model == 'WGAN':
    for p in D.parameters():
        # print(p.data)
        p.data.clamp_(-args.CLAMP, args.CLAMP)
```

3.4 WGAN-GP 神经网络

在实际训练过程中，WGAN 直接使用截断(clipping)的方式来防止梯度过大或过小。但

这个方式太过生硬，在实际应用中仍会出现问题，所以后来产生了其升级版 WGAN-GP.

WGAN-GP 中的 GP 梯度惩罚（gradient penalty）的意思，是替换 weight clipping 的一种方法。通过直接设置一个额外的梯度惩罚项来实现判别器的梯度不超过 k。

其损失函数公式为：

```
if args.model == 'WGAN-GP':
    loss_D += 0.2 * gradient_penalty(D, x_real, x_fake.detach(), batchsz, args)
```

3.5 优化器选择

不同的 GAN 模型可能需要不同的优化器和参数。在原始的 GAN 中，通常使用 Adam 优化器，其参数 betas=(0.5, 0.999)可以使生成器更快地适应判别器的变化。在 WGAN 中，作者建议使用 RMSProp 优化器代替 Adam，因为 Adam 的动量会影响收敛方向。在 WGAN-GP 中，作者又重新使用了 Adam 优化器，其参数 betas=(0.0, 0.9)可以使生成器更稳定地收敛。

参考资料：

(1) [2204.12527] Application of WGAN-GP in recommendation and Questioning the relevance of GAN-based approaches - arXiv.org. https://arxiv.org/abs/2204.12527.

(2) GAN — Wasserstein GAN & WGAN-GP. Training GAN is hard. Models may never… | by Jonathan Hui - Medium. https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490.

(3) Keras documentation: WGAN-GP overriding `Model.train_step`. https://keras.io/examples/generative/wgan_gp/.

(4) Anime Faces with WGAN and WGAN-GP - PyImageSearch. https://pyimagesearch.com/2022/02/07/anime-faces-with-wgan-and-wgan-gp/.

3.6 隐空间语义方向搜索

解如下方程，所需的方向向量 n：

$$2\mathbf{A}^T\mathbf{A}\mathbf{n}_i - 2\lambda_i\mathbf{n}_i = 0.$$

这个过程即求解 A^TA 的特征向量的过程，用如下代码能够实现。

```
################
# TODO factorize the weight of layer0 to get the directions
# run: python sefa.py pggan_celebahq1024 --cuda false/true

_, directions = torch.eig(torch.mm(weight, weight.T), eigenvectors=True)
directions = directions.to(torch.complex64) # 转换为复数类型
directions = directions.transpose(1, 0).real # 获取实部
```

其中，由于 torch 版本较低(1.8.0)，首先计算出矩阵复数类型的特征值和特征向量，然后再取其实部进行求解。

四、 实验结果与分析

4.1 拟合给定分布

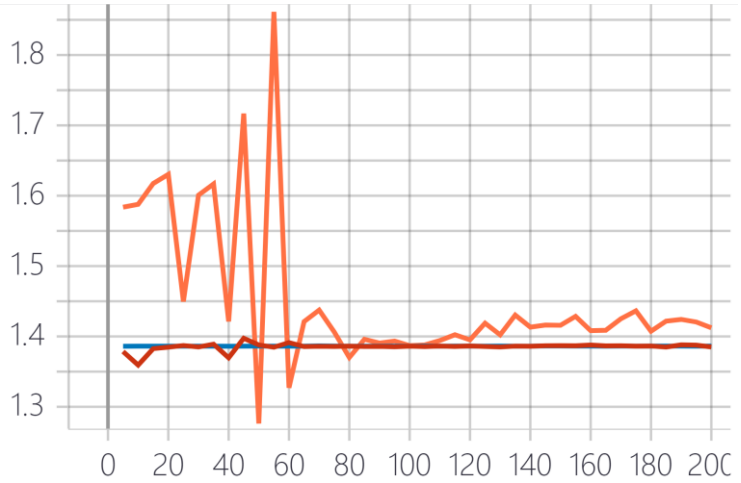
实验基本设置：

训练轮数（Epoch）：200

训练设备：GPU

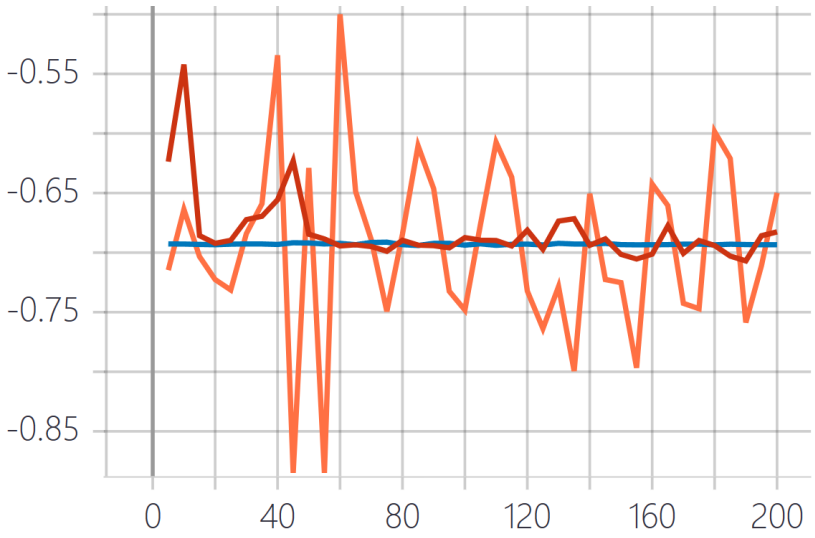
Batch size: 512

训练判别器和生成器损失对比（左图 Loss_D，右图 Loss_G）：



Name	Smoothed	Value	Step	Time	Relative
GAN	1.385	1.385	200	Wed May 31, 00:50:13	15m 30s
WGAN	1.386	1.386	200	Wed May 31, 00:32:01	15m 34s
WGAN-GP	1.412	1.412	200	Wed May 31, 00:12:14	15m 50s

判别器损失

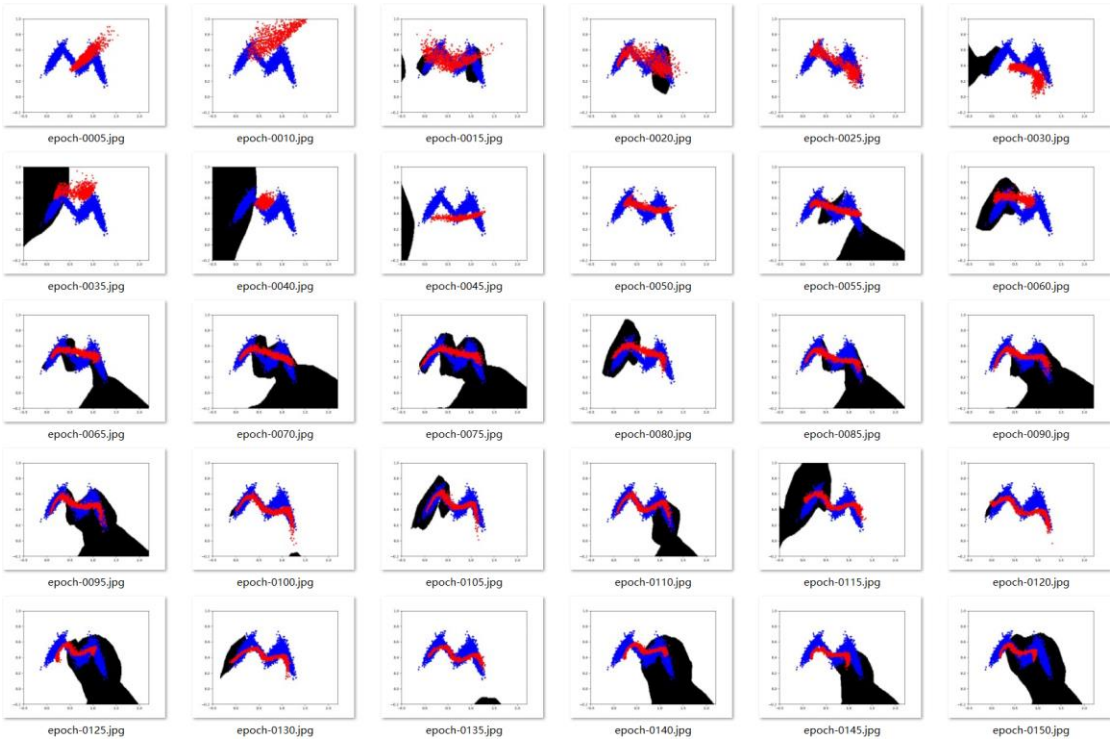


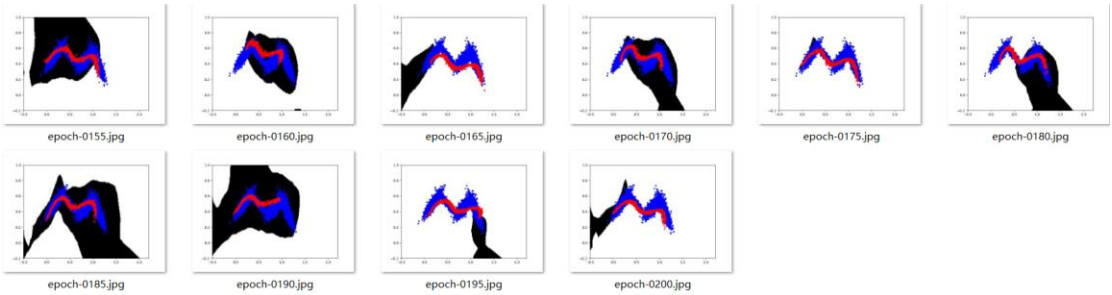
	Name	Smoothed	Value	Step	Time	Relative
	GAN	-0.6826	-0.6826	200	Wed May 31, 00:50:13	15m 30s
	WGAN	-0.6934	-0.6934	200	Wed May 31, 00:32:01	15m 34s
	WGAN-GP	-0.6499	-0.6499	200	Wed May 31, 00:12:14	15m 50s

生成器损失

4.1.1 GAN

学习率：5e-4





Epoch=0-200 时的训练效果可视化

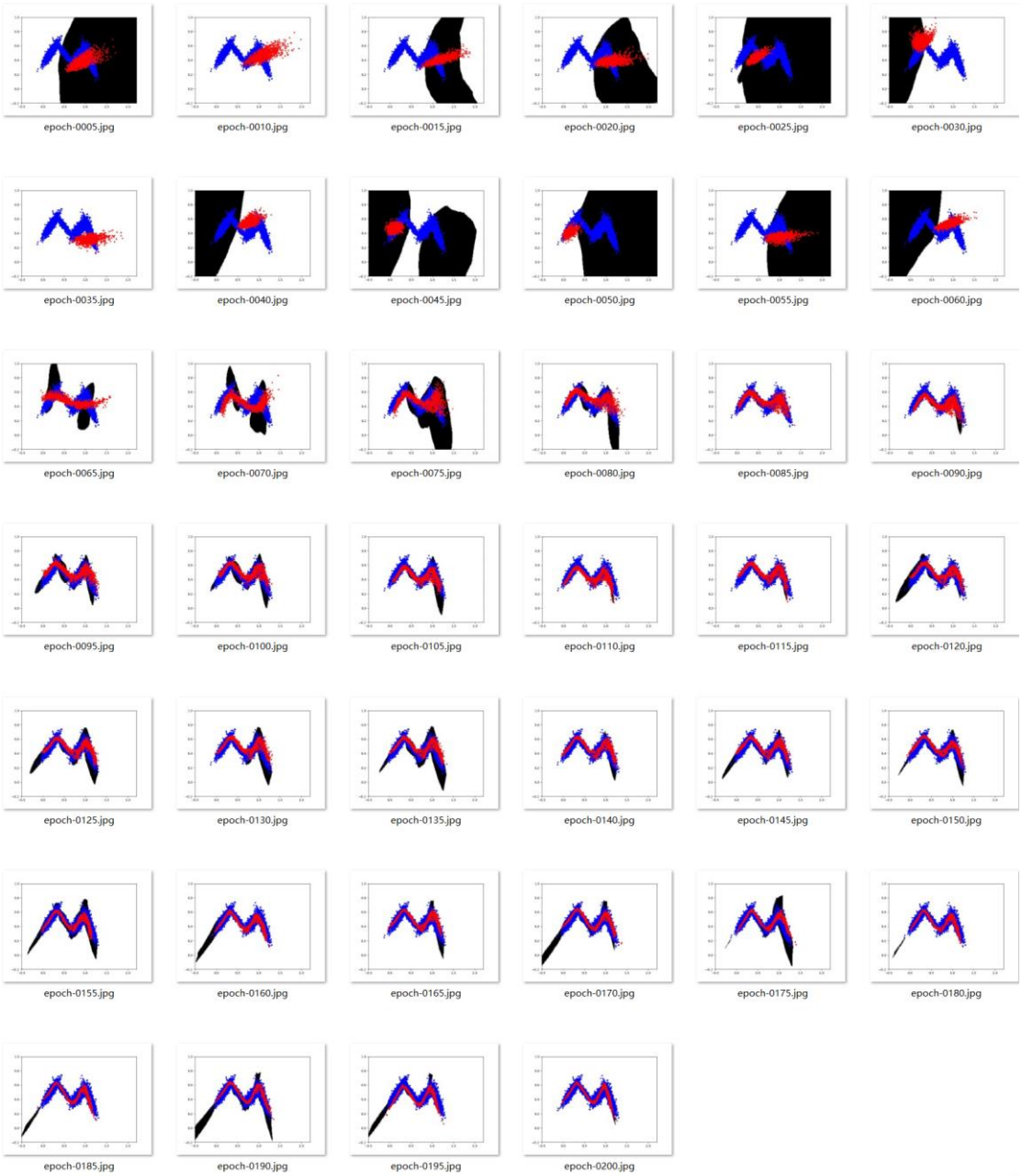
4.1.2 WGAN

学习率：3e-4

Epoch=0-200 时的训练效果可视化

4.1.2 WGAN-GP

学习率：5e-4

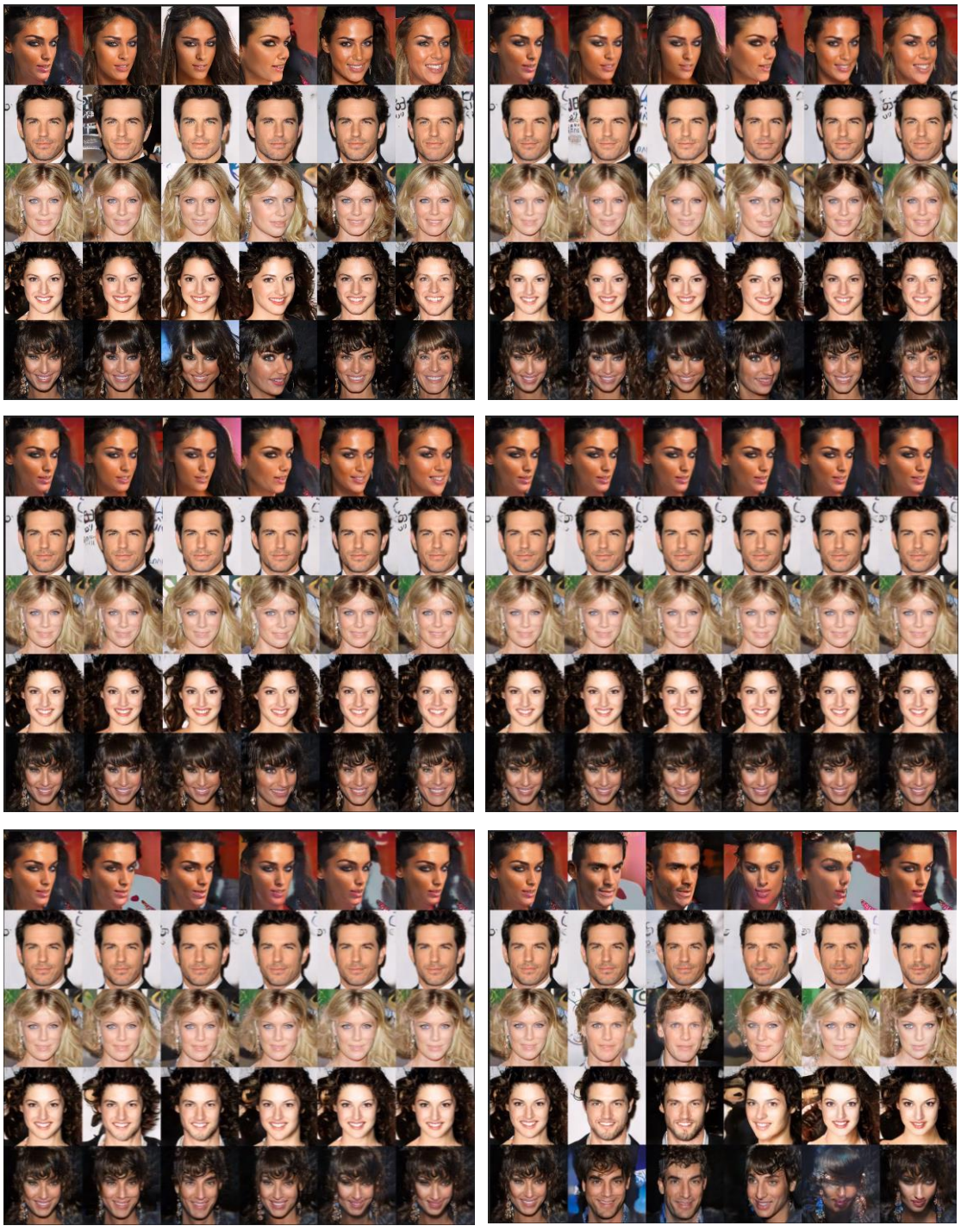


Epoch=0-200 时的训练效果可视化

从中可以看出，WGAN 拟合给定分布的速度最快，WGAN-GP 拟合给定分布的效果最好，GAN 的效果不如 WGAN 和 WGAN-GP。此外，在训练过程中，GAN 的损失函数变化显然范围更大，更不稳定，WGAN-GP 损失变化较小，WGAN 损失变化幅度最小。

此外，生成器和判别器的损失函数在对抗过程中都是震荡的，直到最后生成器和判别器都达到纳什均衡状态，损失函数趋于稳定，波动幅度越来越小。

4.1 隐空间语义方向搜索



图中的变化表示：在隐空间当中，选择特征值最大的 k 个特征向量作为方向，随后给定步长，将其加到 Generator 的输入向量 z 中进行训练，最终就可以得到特定语义对于图像的变换。

其中，第一列未变化，第二列性别变为男性，第三列性别和背景改变，第四列角度改变，第五列容貌、头发等装饰细节改变，第五列容貌和表情改变。

五、 结语

通过该实验，我体会到了生成式对抗网络对博弈论思想的应用，和拟合分布的生成实践中的有效性。以及通过人脸图片切身体会到了，隐空间方向所代表的实际语义，体会了向量的数学世界到感知世界的联系。