哈尔滨工业大学

模式识别与深度学习

实验四：生成式对抗网络

姓　　名：刘天瑞

院　（系）：未来技术学院

专　　业：视听觉信息处理

学　　号：7203610121

指导教师：左旺孟

提交日期：2023.5.26

摘 要

本次深度学习实验为模式识别与深度学习课程的实验五：即生成式对抗网络。在本次实验中，我利用 Python 中的 Pytorch 库从而自己实现了 GAN，WGAN 以及 WGAN-GP 共三种生成式对抗网络的具体结构，继而拟合了 points.mat 中的数据分布，然后还对比了上述这三种模型的实验效果。除此之外，我基于课程所给定的 ProGAN 相关代码和具体模型下载路径，也成功实现了 ProGAN 模型的 SeFa 部分，从而完成了隐空间语义方向搜索的任务。

关键词：模式识别与深度学习，生成式对抗网络，GAN，WGAN，WGAN-GP，ProGAN

目 录

一、 深度学习框架与实验环境

本次实验采用的深度学习框架是 Python 中强大的深度学习库 Pytorch，整个实验是在 Visual Studio + Pip 环境下完成的。Pip 是一个开源的 Python 发行版本，可以很方便地安装许多深度学习所需要的模块包，而 Visual Studio 则是一个功能强大的 IDE，可以在其中完成 python 代码编写深度学习过程、进行训练测试等深度学习环节。由于我在大二时参加过美赛有接触到一些深度学习相关的工具知识，因此在本次实验中的配置环境过程相对得心应手。配置环境的具体流程比较繁琐，查看系统 Pytorch 库以及 cuda 版本如下图 1 所示（英伟达表示 cuda 可升级到的最高版本）：

```
C:\Users\刘天瑞>python
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
2.0.0+cu118
>>> ^Z
```

```
>>> import torch
>>> print(torch.backends.cudnn.version())
8700
```

```
C:\Users\刘天瑞>nvidia-smi
Sat Apr 29 18:24:48 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 531.41                 Driver Version: 531.41       CUDA Version: 12.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf            Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 3060 L...  WDDM | 00000000:01:00.0  On |                  N/A |
| N/A   39C    P8              15W /  N/A|   1491MiB /  6144MiB |     21%      Default |
|                                         |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      2884    C+G   ...on\112.0.1722.58\msedgewebview2.exe    N/A      |
|    0   N/A  N/A      4944    C+G   ...crosoft\Edge\Application\msedge.exe    N/A      |
|    0   N/A  N/A      5324    C+G   ...\cef\cef.win7x64\steamwebhelper.exe    N/A      |
|    0   N/A  N/A      9704    C+G   ...7.0_x64__w2gh52qy24etm\Nahimic3.exe    N/A      |
|    0   N/A  N/A      9900    C+G   C:\Windows\explorer.exe                   N/A      |
|    0   N/A  N/A     10908    C+G   ...nt.CBS_cw5n1h2txyewy\SearchHost.exe    N/A      |
|    0   N/A  N/A     11620    C+G   ...5n1h2txyewy\ShellExperienceHost.exe    N/A      |
|    0   N/A  N/A     13316    C+G   ...2txyewy\StartMenuExperienceHost.exe    N/A      |
|    0   N/A  N/A     16712    C+G   ...CBS_cw5n1h2txyewy\TextInputHost.exe    N/A      |
|    0   N/A  N/A     17748    C+G   ...GeForce Experience\NVIDIA Share.exe    N/A      |
|    0   N/A  N/A     19364    C+G   ...GeForce Experience\NVIDIA Share.exe    N/A      |
|    0   N/A  N/A     20712    C+G   ...1.0_x64__8wekyb3d8bbwe\Video.UI.exe    N/A      |
|    0   N/A  N/A     22552    C+G   ...t.LockApp_cw5n1h2txyewy\LockApp.exe    N/A      |
|    0   N/A  N/A     25060    C+G   ...4036\office6\promecefpluginhost.exe    N/A      |
|    0   N/A  N/A     25076    C+G   ...__8wekyb3d8bbwe\WindowsTerminal.exe    N/A      |
|    0   N/A  N/A     25856    C+G   ...siveControlPanel\SystemSettings.exe    N/A      |
|    0   N/A  N/A     26908      C   ...Programs\Python\Python39\python.exe    N/A      |
|    0   N/A  N/A     26948    C+G   ...__8wekyb3d8bbwe\WindowsTerminal.exe    N/A      |
|    0   N/A  N/A     30532    C+G   ...22\Community\Common7\IDE\devenv.exe    N/A      |
|    0   N/A  N/A     31092    C+G   ...ft Office\root\Office16\WINWORD.EXE    N/A      |
|    0   N/A  N/A     31864    C+G   ...4036\office6\promecefpluginhost.exe    N/A      |
+-----------------------------------------------------------------------------+
```

图 1

二、 生成式对抗网络实现（分布拟合任务）

2.1 三种网络结构的实现

在本次深度学习实验中，我利用 Python 中的 Pytorch 成功地实现了包括 GAN，WGAN 以及 WGAN-GP 总共三种生成式对抗网络的具体结构（我将其具体运行代码分别保存在根目录下的 GAN.py，WGAN.py 以及 WGAN_GP.py 程序文件中）。其中以 GAN 网络为例，它的生成器 Generator 和判别器 Discriminator 结构模块如下图 1 和图 2 所示：

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(input_size, 64),
            nn.LeakyReLU(0.2, inplace = True),
            nn.Linear(64, 256),
            nn.BatchNorm1d(256, 0.8),
            nn.LeakyReLU(0.2, inplace = True),
            nn.Linear(256, 512),
            nn.BatchNorm1d(512, 0.8),
            nn.LeakyReLU(0.2, inplace = True),
            nn.Linear(512, 2)
        )

    def Forward(self, x):
        return self.net(x)
```

图 1

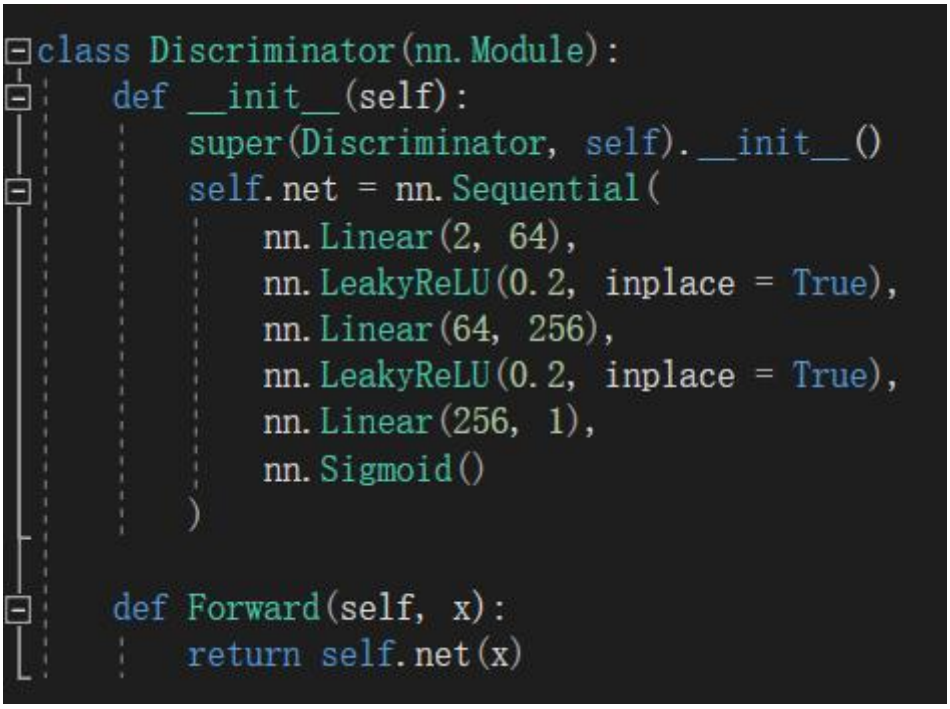


图 2

而相较于普通的 GAN 网络，WGAN 网络主要有如下所示 4 方面来进行改进：

1. 去掉了判别器 Discriminator 的最后一层：Sigmoid；

2. 将判别器 Discriminator 的 w 取值限制在了[-c,c]的区间之内，从而确保了 lipschitz 连续；

3. 使用不带有 log()函数的 loss 损失值计算；

4. 不使用具有动量的优化方法（例如 Adam 优化器）。

而 WGAN-GP 网络相较于 WGAN 网络，则又引入了梯度惩罚项。

2.2 网络模型训练过程

在本次深度学习实验中，我首先读取了拟合分布页 points.mat 的’xx’维度，并同时将样本随机打乱顺序，然后再随机提取其中的 7000 个点（point）进行训练（即训练集大小）；而剩余的 1192 个点我就用来画数据分布散点图，从而来验证生成器输出的数据是否拟合该分布。

除此之外，在训练数据时，我将 batch_size 设定为 256，学习率则设定为 0.00005，生成器所输入的噪声维度设定为 2，优化器则均采用了 RMSprop，最后总共进行了 320 轮 epoch 的训练，其实验结果如下一小节所示：

2.3 GAN 网络训练结果

GAN 网络训练结果如下图 3 所示：

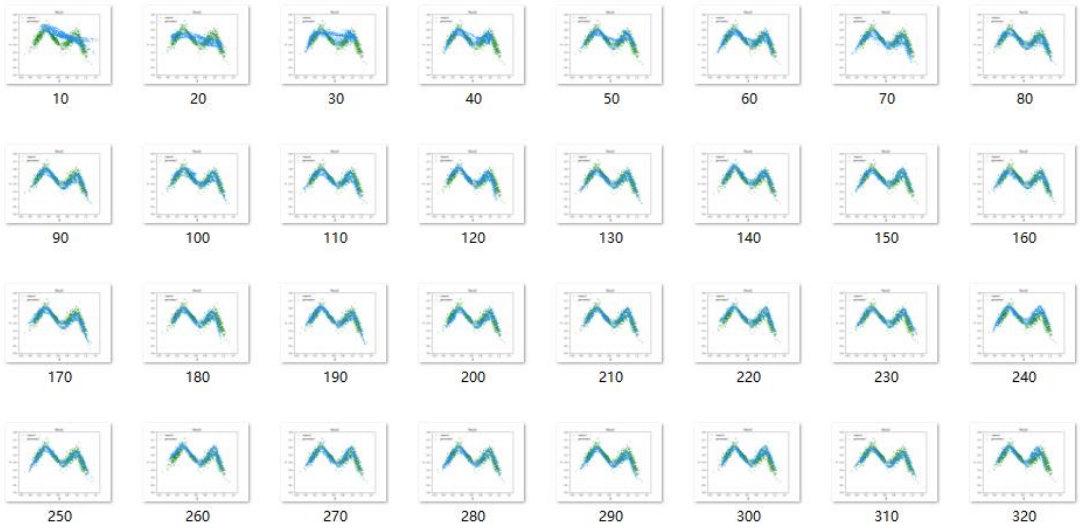


图 3

经过 320 轮 epoch 之后的 GAN 网络训练结果如下图 4 所示：

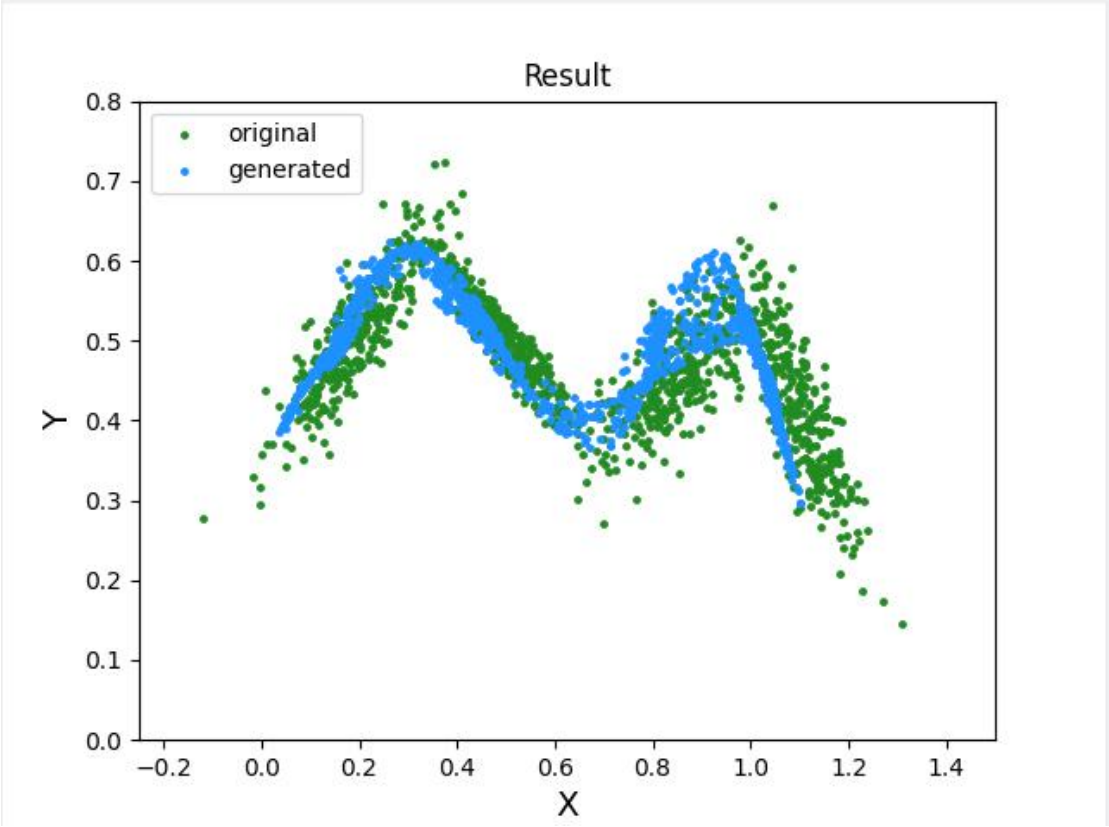


图 4

从上述实验结果可以初步得出结论，即 GAN 网络结构的训练效果较好，并且是在训练了大约 50 轮 epoch 之后就已经有了较好的效果，最终比较之下页能够保持稳定波动。

2.4 WGAN 网络训练结果

WGAN 网络训练结果如下图 5 所示：

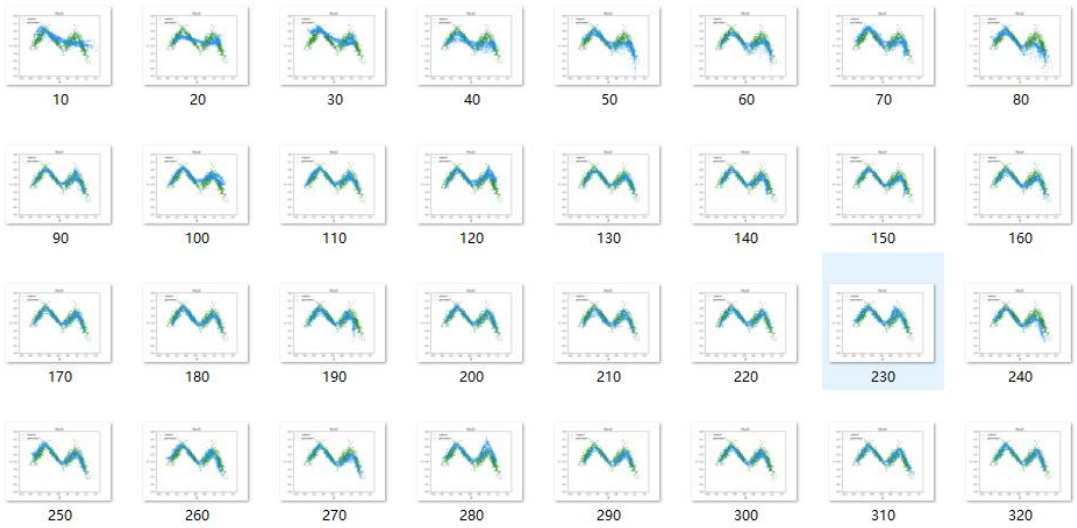


图 5

经过 320 轮 epoch 之后的 WGAN 网络训练结果如下图 6 所示：

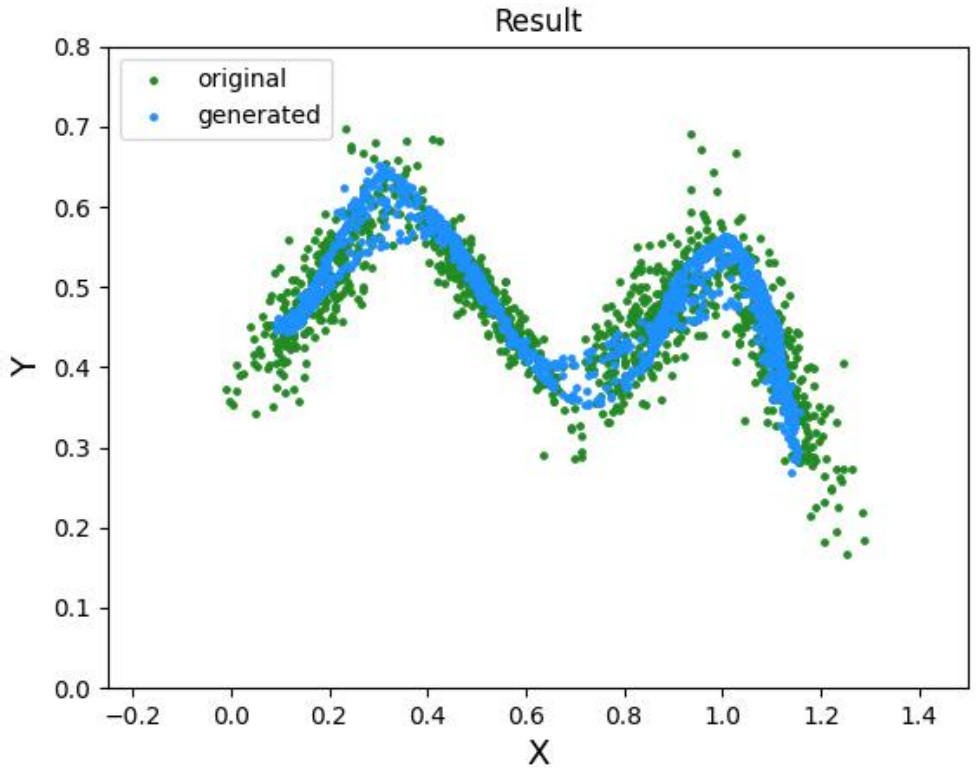


图 6

从上述实验结果可以初步得出结论，即 WGAN 网络结构的训练效果与 GAN 网络类似，都较为优秀，并且也是在训练几十轮 epoch 之后就已经有了较好的效果，最终比较之下亦能够保持稳定波动。

2.5 WGAN-GP 网络训练结果

WGAN-GP 网络训练结果如下图 7 所示：

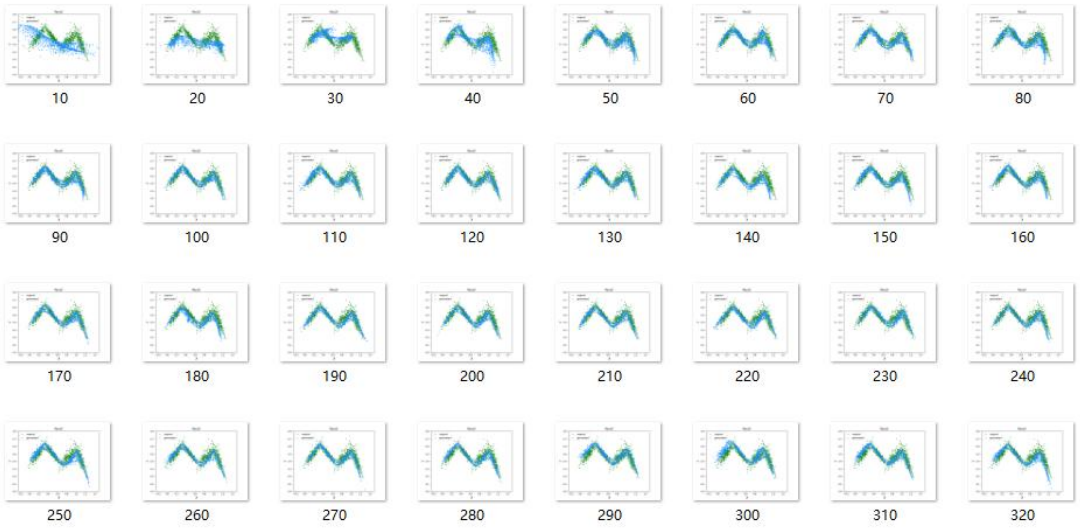


图 7

经过 320 轮 epoch 之后的 WGAN-GP 网络训练结果如下图 8 所示：

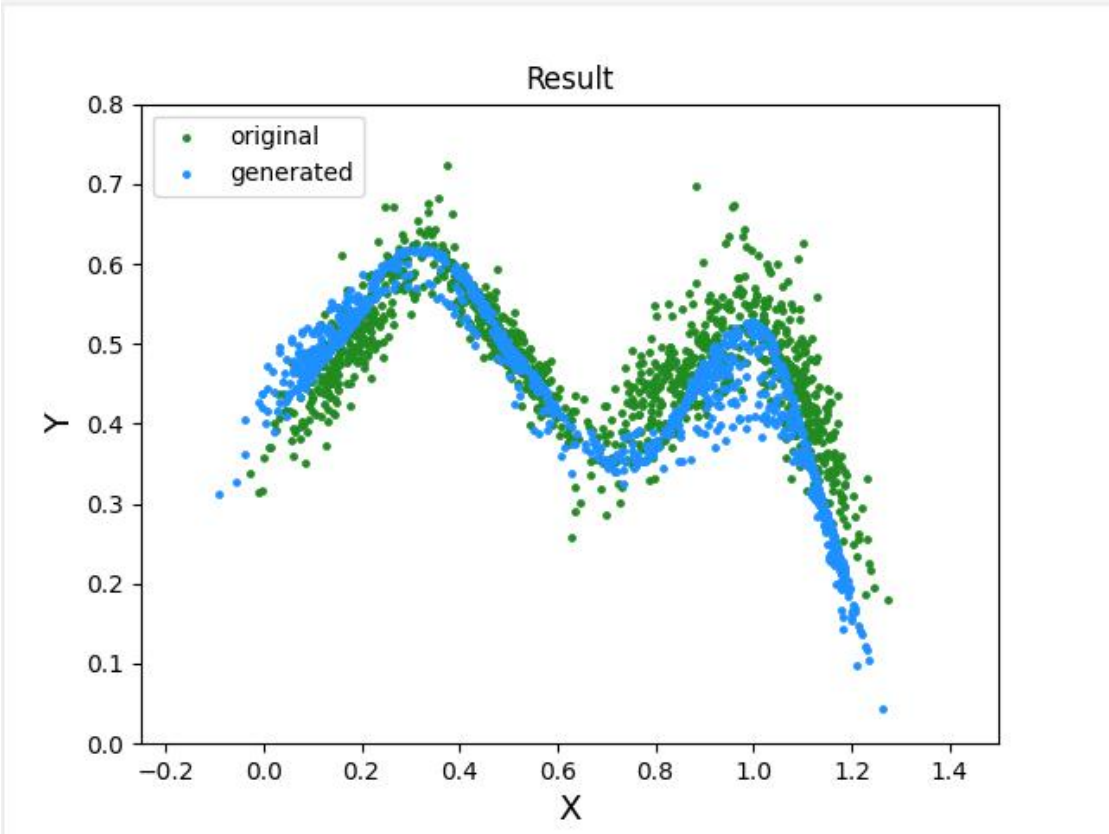


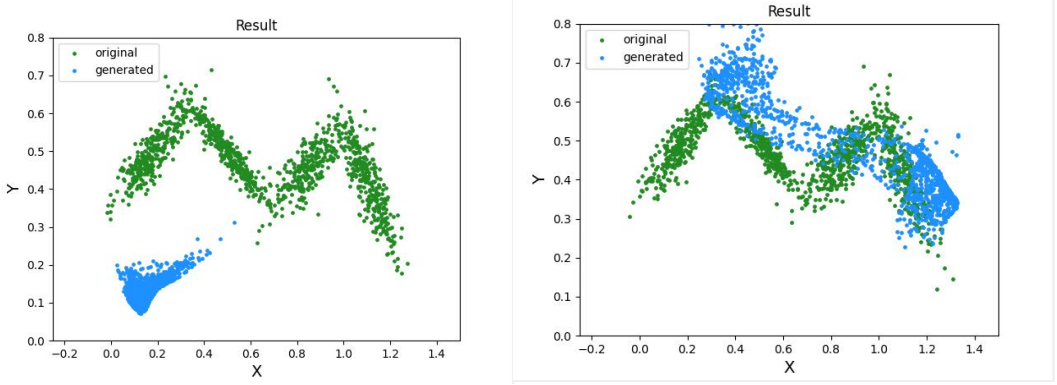
图 8

从上述实验结果可以初步得出结论，即 WGAN-GP 网络结构的训练效果明

显相较于其他两种收敛较慢，可以发现在训练了大约 200 轮 epoch 之后拟合训练得出的散点才能较为贴合原始数据的分布。

2.6 更换优化器对比不同影响

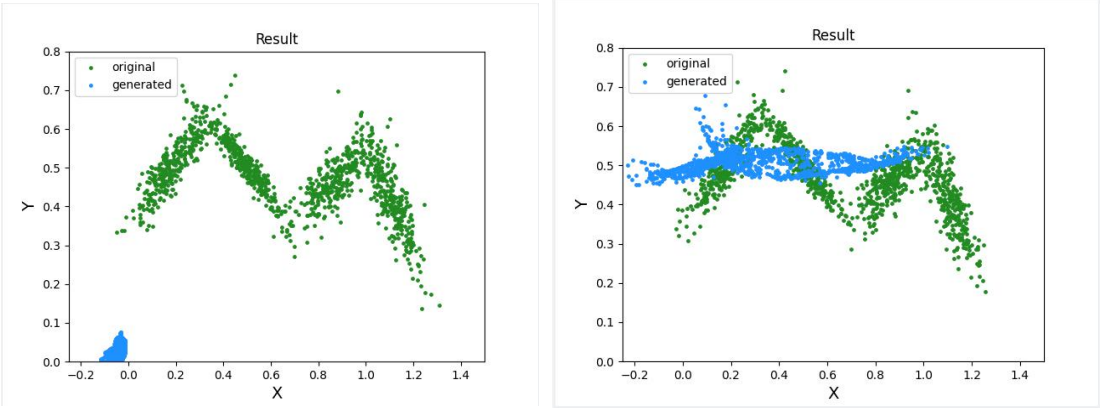
使用 GAN 网络结构模型，将优化器分别更换为 SGD 与 Adam 不同优化器，训练 320 轮 epoch 之后对比其实验拟合分布效果如下图 9 所示：



（1）使用 SGD　　（2）使用 Adam

图 9

使用 WGAN 网络结构模型，将优化器分别更换为 SGD 与 Adam 不同优化器，训练 320 轮 epoch 之后对比其实验拟合分布效果如下图 10 所示：

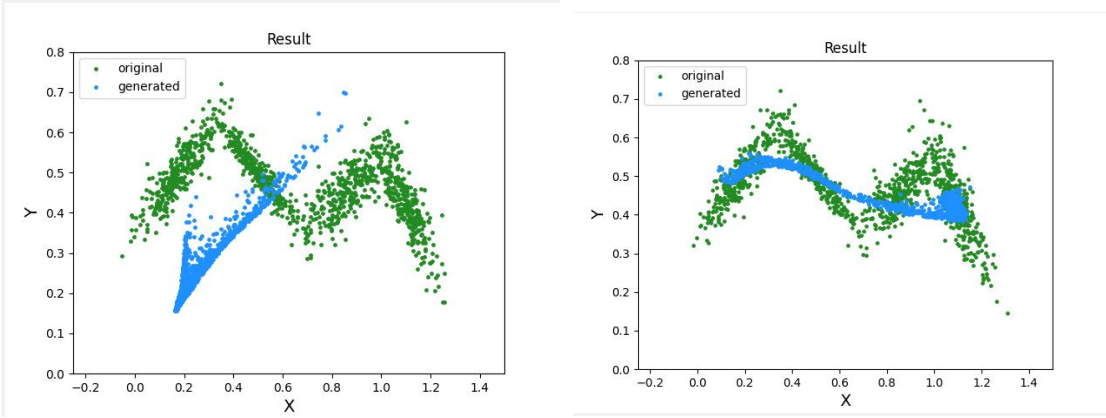


（1）使用 SGD　　（2）使用 Adam

图 10

使用 WGAN-GP 网络结构模型，将优化器分别更换为 SGD 与 Adam 不同优

化器，训练 320 轮 epoch 之后对比其实验拟合分布效果如下图 11 所示：



（1）使用 SGD　　（2）使用 Adam

图 11

从上述三个网络模型的实验对比拟合效果可以明显看出，换用 SGD 优化器之后，拟合图形几乎从开始到结束都没有明显移动，这说明模型训练速度变得很慢；而换用 Adam 优化器之后拟合图形则过程变化波动非常大，具有十分明显的不稳定性。由此可以得出这两种优化器的优化效果均不如实验初期使用的 RMSprop 优化器。

2.7　实验结果对比分析

综上所述，这三种网络结构模型在使用 RMSprop 优化器时，都能在 320 轮 epoch 内得到较好的拟合效果，其中，WGAN-GP 网络结构模型的速度相较于其他两个网络模型明显稍慢，但其拟合输出的散点分布更为均匀（具体表现为拟合图形更加纤细密集）。而换用其他优化器则很难得到如此较好的效果。

最终实验结果的相关动图文件.gif 制作可以运行 DRAW.py 程序文件来生成，我将其保存在了./GIF 目录文件夹下。

三、隐空间语义方向搜索

3.1　主要研究内容及背景知识

GAN 网络结构模型中的生成器通常以随机采样的潜在向量作为输入，生成高保真图像。通过改变潜在向量 z，我们可以改变输出图像,例如头发颜色、面部表情、姿势、性别等，我们需要知道移动潜在向量 z 是什么。

对于 GAN 网络结构模型的每一层，都学习从一个空间到另一个空间的转换。对于第一层变换，总有 G(z)=Az+b 满足条件，而给定向量 z 一个方向 n 和步长α，总有 G(z+αn)=G(z)+αAn，由此如果给定一个潜在码 z 和方向向量 n，则可以通过在变换后的投影码上加αAn 来实现对图像编辑的过程。这里可以看出 A 中包含图像变换的基本信息，所以我们只需要对 A 进行分解就可能会发现特定图像变换的方向信息。

根据论文中的方法，我们求出 A^TA 的特征向量，要找出 k 个使图像进行变换的信息，即选择特征值最大的 k 个特征向量作为方向。随后给定一定的步长，将其加到生成器 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。

最终根据生成的图像和视频，每一列对应的语义为性别，性别，视角的变化，头发和年龄。

3.2 sefa.py 相关代码补全与模型实现

在这一模块中，我补全了 sefa.py 中的相关代码缺漏部分，其实主要完成的任务就是分解 layer0 的权重，从而得到了各所有方向 directions 的参数继而找到对应语义含义。具体的补全运行代码部分如下图 12 所示：

```
################
# TODO factorize the weight of layer0 to get the directions
# run: python sefa.py pggan_celebahq1024 --cuda false/true
_, directions = torch.linalg.eig(torch.mm(weight, weight.T))

directions = directions.transpose(1, 0).real
################
```

图 12

除此之外，在相关代码补全完成之后，需要自己手动下载 model.zoo.py 程序文件中的 pggan_celebahq1024 模型文件到目录文件夹 checkpoints 中，然后在分析器构建输入的参数中添加模型名称：default="pggan_celebahq1024"，最后可以成功运行 sefa.py 程序，模型训练之后得到如下图 13 所示的实验结果：

图 13

将视频结果保存在了./work_dirs/synthesis/pggan_celebahq1024_N5_K5_seed0 目录下。

最后说明一下文档归类，本次深度学习实验共由两部分组成，其中生成式对抗网络实现的数据拟合相关文件保存在了./GAN 目录下，隐空间语义方向搜索相的关文件保则存在了./genforce 目录下。

源代码以及注释见附件。