南開大學
Nankai University

网络空间安全学院

深度学习实验报告

实验四：生成对抗网络

姓名：武桐西

学号：2112515

专业：信息安全

指导教师：侯淇彬

2024 年 6 月 25 日

目录

1 实验目的

本次实验的主要内容是生成对抗网络（Generative Adversarial Network, GAN）。通过本次实验，需要掌握以下内容：

1. 掌握生成对抗网络（GAN）的基本原理。
2. 学会使用 PyTorch 搭建 GAN 网络来训练 FashionMNIST 数据集。

本次实验的总揽如下：

1. 实现 GAN 网络，并在 FashionMNIST 数据集上训练和测试。
2. 采用卷积实现 GAN 网络，并在 FashionMNIST 数据集上训练和测试。
3. 探究隐变量对 GAN 图像生成的影响。

2 实验环境

本实验在 macOS 系统下测试，搭配 M3 Max 芯片，使用 Python`3.11.5`解释器，PyTorch 版本为`2.2.0`（使用 MPS 进行硬件加速），如表1所示。

表 1: 实验环境

操作系统	macOS	硬件	M3 Max (支持 MPS)
Python	3.11.5	PyTorch	2.2.0 (支持 MPS)

本次实验的代码已上传GitHub。

3 文件目录结构

本次实验的文件目录结构如下：

```
GAN
├── data ..... Data Path
├── models ..... Models
│   ├── cnn_gan.py ..... CNN-Based GAN
│   └── gan.py ..... GAN
├── user_interact.ipynb ..... 用户交互：探究隐变量的影响
├── utils.py ..... 训练 GAN、图像生成、结果可视化
└── main.py ..... Main Script
```

4 实验原理与实验过程

在本部分，首先介绍本次实验的目标任务和数据集，然后介绍三种前馈神经网络架构的实现，最后介绍整个实验的流程。

4.1 目标任务

本实验需要实现**图像生成**任务，使用 FashionMNIST 数据集进行训练，该数据集中的图像为灰度图像（只有一个通道），其形状为 28×28。

4.2 GAN

基础 GAN 网络 [1] 的网络结构如代码1所示，其基本架构采用 MLP（全连接层），激活函数采用的是 LeakyReLU，而不是在 CNN 中经常用到的 ReLU，同时使用批归一化 BatchNorm 等正则化方法抑制过拟合。

代码 1: GAN Model Architecture

```
Discriminator(
  (model): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=784, out_features=512, bias=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
    (3): Linear(in_features=512, out_features=256, bias=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Linear(in_features=256, out_features=1, bias=True)
    (6): Sigmoid()
  )
)
Generator(
  (model): Sequential(
    (0): Linear(in_features=100, out_features=256, bias=True)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): BatchNorm1d(256, eps=0.8, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (3): Linear(in_features=256, out_features=512, bias=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): BatchNorm1d(512, eps=0.8, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (6): Linear(in_features=512, out_features=784, bias=True)
    (7): Tanh()
  )
)
```

4.3 CNN-Based GAN

基于 CNN 的 GAN 网络 [2] 的结构如代码2所示。

代码 2: CNN-Based GAN Model Architecture

```
Discriminator(
  (model): Sequential(
    (0): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(1, 1))
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (8): Conv2d(256, 1, kernel_size=(4, 4), stride=(1, 1))
    (9): Sigmoid()
  )
)
Generator(
  (linear): Sequential(
    (0): Linear(in_features=100, out_features=6272, bias=True)
  )
  (model): Sequential(
    (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (1): Upsample(scale_factor=2.0, mode='nearest')
    (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Upsample(scale_factor=2.0, mode='nearest')
    (6): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True,
       ↪ track_running_stats=True)
    (8): LeakyReLU(negative_slope=0.2, inplace=True)
    (9): Conv2d(64, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): Tanh()
  )
)
```

在判别器中，将基础的 GAN 的全连接层替换为卷积层；在生成器中，采用上采样`Upsample`和卷积操作为基本算子。

4.4 实验流程

本次实验的实验流程如下：

1. **数据预处理**：加载 FashionMNIST 数据集，并进行数据预处理（转化为张量形式、数据归一化等）。
2. **模型选择**：根据命令行参数的解析选择要使用的模型。
3. **模型训练**：利用 Min-Max 算法的思想，采用梯度优化的方式进行模型训练。
4. **模型评估**：生成器将噪音转换为图像，以便人为进行视觉检查和模型评估。
5. **结果可视化**：绘制训练过程中的损失以及判别器对真实图像和生成图像的输出概率。

5 实验结果与分析

5.1 实验设置

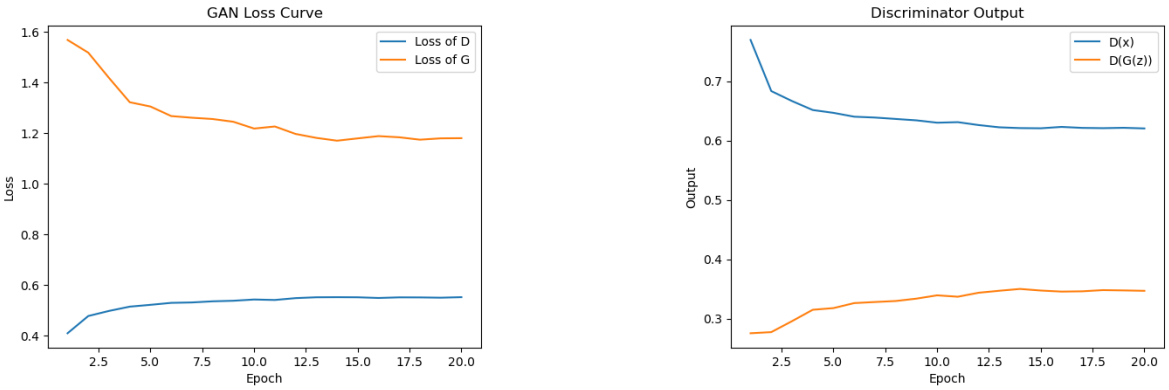
本实验中，所有的超参数设置均相同，如表2所示。

表 2: 实验设置

batch size	64	# of epoch	20
loss function	BCELoss	learning rate	0.0002
优化器	Adam	betas	(0.5, 0.999)
硬件加速	MPS		

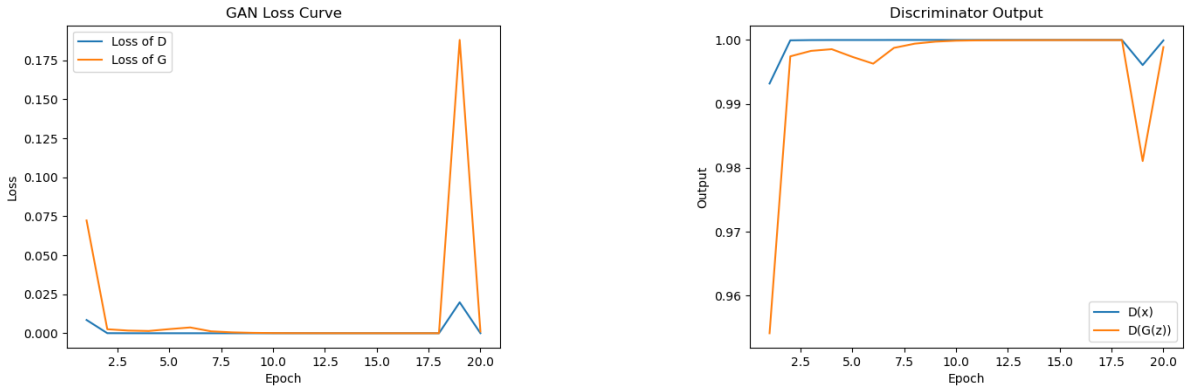
5.2 结果与分析

基础的 GAN 网络（GAN）和基于卷积的 GAN（CNN-Based GAN）分别如图5.1和图5.2所示。



(a) 损失曲线 (b) 判别器输出概率-Epoch 曲线

图 5.1: GAN Model 结果



(a) 损失曲线 (b) 判别器输出概率-Epoch 曲线

图 5.2: CNN-Based GAN Model 结果

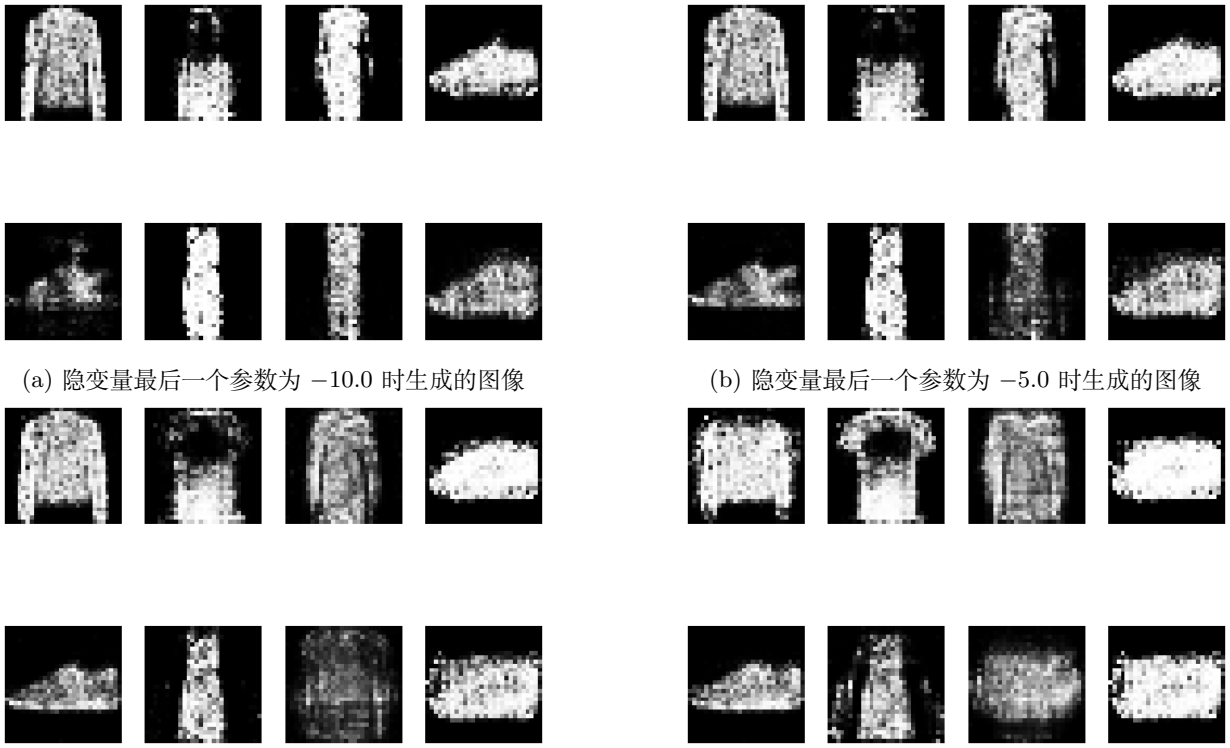
GAN 与 CNN-Based GAN 模型生成的图像分别如图5.3a和图5.3b所示。

(a) GAN 模型生成的示例图像 (b) CNN-Based GAN 模型生成的示例图像

图 5.3: 两种 GAN 模型生成的示例图像

从图中可知，GAN 模型非常出色地完成了图像生成任务；而 CNN-Based GAN 由于模型参数量过大、模型复杂度过高、数据集比较简单而导致严重的**过拟合现象**。

5.3 隐变量对 GAN 图像生成的影响研究



(a) 隐变量最后一个参数为 −10.0 时生成的图像

(b) 隐变量最后一个参数为 −5.0 时生成的图像

(c) 隐变量最后一个参数为 0.0 时生成的图像

(d) 隐变量最后一个参数为 5.0 时生成的图像

图 5.4: 隐变量对 GAN 图像生成的影响

由于篇幅所限，这里不过多展示。以 100 维的隐变量的最后一个参数为例，当其分别为 −10.0、−5.0、0.0、5.0 时，GAN 生成的图像的变化如图5.4所示。可以观察到，隐变量会影响生成的图像的质量，比如噪声点的分布情况，由于 FashionMNIST 图像本身的分辨率较低，因此生成的图像的噪声点可能不太明显。

6 总结与体会

在本次实验中，笔者深入了解生成对抗网络的原理，并编程实现基础的 GAN 网络和基于卷积的 GAN 网络，并对梯度优化方法和 Min-Max 博弈有了更深刻的认识。

参考文献

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[2] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[3] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.