# 《人工智能导论》大作业

任务名称：Mnist 条件生成器

完成组号：　　/　　

小组人员：李毅恒、刘宇航、刘晨宇、邱俊豪、杨天沐

完成时间：2023 年 6 月 17 日

# 1、任务目标

MNIST（Modified National Institute of Standards and Technology）条件生成器的任务目标是通过给定手写数字图像的标签，来生成相应的手写数字图像。其主要包含以下两个方面的任务：

（1）图像识别：模型需要了解每个数字的特征和形态，并能够根据这些特征和形态来判断输入标签对应的数字是什么。

（2）图像生成：模型需要学习到 MNIST 数据集中各数字的特征分布，以及不同标签与数字之间的对应关系，进而生成与给定标签相对应的手写数字图像。即，在已知标签情况下，生成具有该标签特征的高质量手写数字图像。

本次大作业基于 MNIST 数据集，构建一个条件生成模型，输入的条件为 0~9 的数字，输出对应条件的生成图像。要求：

（1）支持随机产生输出图像；

（2）在 cpu 上有合理的运行时间。

# 2、具体内容

## （1）实施方案

**①数据预处理：**下载 MNIST 数据集，并将其分为训练集和测试集。设置输入条件为 0—9，对图像进行标准化和归一化处理，以便于后续处理。

**②构建生成器模型：**生成器模型是一个神经网络，它将随机向量和条件作为输入，生成类似于 MNIST 数据集的图像。本组使用 GAN 来构建生成器模型。

**③构建判别器模型：**判别器模型是一个神经网络，它将图像和条件作为输入，预测图像是否来自 MNIST 数据集。

**④训练模型：**使用训练集对生成器和判别器进行训练，通过最小化损失函数来优化模型。在训练过程中，使用梯度下降等优化算法来更新模型的参数。

**⑤生成图像：**使用训练好的生成器模型和随机向量作为输入，可以生成类似于 MNIST 数据集的图像。

**⑥评估模型：**使用测试集评估模型的性能，可以计算模型的准确率、精度、召回率等指标来评估模型的效果。

**⑦调整模型：**根据评估结果，调整模型的参数、网络结构等来提高模型的性能。

## （2）核心代码分析

①导入所需的库和模块。

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt
import torchvision
from torchvision import transforms
from torch.utils import data
import os
import glob
from PIL import Image

import time
```

图 1 导入的库与模块

②定义了一个名为 AiGcMn 的类，用于封装 GAN 的实现。它包含了嵌套类 Generator 和 Discriminator，分别定义了生成器和判别器网络。

```
class AiGcMn:
    def one_hot(self, x):
        return torch.eye(10)[x, :]

    class Generator(nn.Module):
        def __init__(self):
            super().__init__()
            # 生成器网络层的定义

    class Discriminator(nn.Module):
        def __init__(self):
            super().__init__()
            # 判别器网络层的定义

    def __init__(self):
```

图 2 AiGcMn 类的定义

③generate_and_save_images 函数，接收一个经过训练的模型、当前的训练轮数、标签输入和噪声输入，并使用生成器模型生成一组图像。这些图像随后被显示和保存以进行可视化。

```python
def generate_and_save_images(self, model, epoch, label_input, noise_input):
    # 生成并保存图像

def generate(self):
    # 训练循环
    for epoch in range(50):
        # 迭代数据集
        for step, (img, label) in enumerate(self.dataloader):
            # 判别器的前向传播和反向传播

            # 生成器的前向传播和反向传播

        # 生成并保存样本图像
```

图 3 generate_and_save_images 函数

④流程分析：

generate 函数是主要的训练循环。

I.首先定义空列表 D_loss 和 G_loss，用于保存每个 epoch 的判别器和生成器的损失。

II.确定权重剪裁的限制值 weight_cliping_limit，用于在训练判别器时对判别器的参数进行剪裁。

III.创建张量 one 和 mone，分别表示正数和负数的标量 1，用于后面的损失计算。

IV.开始主循环，进行指定次数的训练迭代（图 3 所示的是 50 次）。

V.然后计算生成器的损失。如果满足条件 epoch % 1 == 0，调用 self.generate_and_save_images() 方法生成并保存一批图像样本。

VI.最后打印每个 epoch 的训练时间、判别器损失和生成器损失，以及总的训练时间。

## 3、工作总结

### （1）收获、心得

**①熟悉深度学习模型的建立和训练：**通过本次大作业本组同学掌握了如何使用 PyTorch 框架来构建 MNIST 条件生成模型，并了解深度学习模型的基本训练过程。

**②理解图像生成技术：**通过本次大作业本组同学初步了解了生成式对抗网络

（GANs）和条件生成模型等图像生成技术，了解它们是如何工作的，以及如何在实践中应用这些技术。

**③提高编程能力：**通过实现代码，本组同学训练了自己编程能力，同时也会学习到如何编写一个高效的程序。

**④提高数学学科知识：**通过本次作业本组同学掌握了一些重要的数学方法和技巧，例如梯度下降、线性代数和概率论，这些知识对于理解深度学习模型和优化技术非常重要。

**⑤了解图像生成的应用领域：**在这次大作业中，我们除了可以学习到如何生成手写数字图像之外，还可以了解到图像生成技术在其他领域的应用，如自然语言处理、计算机视觉等。

## （2）遇到问题及解决思路

### ①优化方面的问题

**I.判别器训练次数优化**

在 GAN 中，生成器是通过对比与真实样本的区别来进行优化的。为了确保生成器得到良好的训练和提高生成样本的质量，**必须保证判别器具有足够的训练次数和准确性。**

在项目中，我们发现判别器训练轮次在 **40 epoch 的时候的效果比训练 50 epoch 的效果要好一些。**

(a) 迭代次数=40　　(b) 迭代次数=50

图 4 开发前期生成效果图

**II.WGAN 优化**

在搜索优化方法的时候，发现传统 GAN 会有梯度消失的问题。为了解决这

个问题，我们修改了模型，**使用了 WGAN 的训练方法来更新我们的模型参数。**

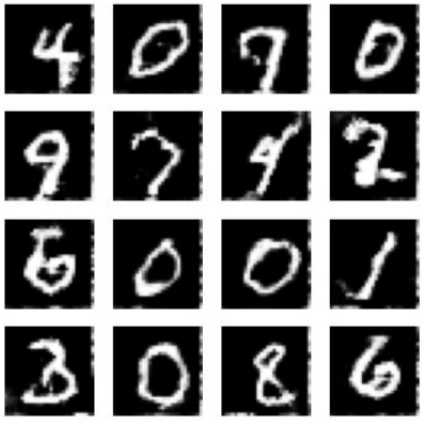

图 5 WGAN 优化后的生成效果图

**②优化后的生成器性能**

在对生成器进行优化后，目前的生成器运行一次迭代的时间约在 1 分钟左右，能够在 25 次迭代次数左右输出比较让人满意的结果，如图 6 所示。

图 6 当前生成器 25 次迭代后对 0~9 各个数字的生成效果图

## 4、课程建议

我们小组认为通过人工智能导论这门课，收获了但不限于人工智能的概念和原理、核心技术以及对社会的影响。与此同时，本组仍有如下课程建议：

**（1）更多实践经验：**学生希望能够获得更多的实践经验，例如通过编写代码和运行模拟项目来加深对课程内容的理解。老师可以结合实际案例和开源项目，指导学生学习和实现自己的 AI 项目。

**（2）注重应用领域：**虽然理论知识很重要，但学生也想了解人工智能在真实世界中的应用。老师可以向学生介绍一些成功的人工智能应用案例，并讨论这些技术如何改变我们的生活和工作。

**（3）组织小组讨论：**人工智能是一个涵盖众多领域的复杂主题，不同学生

可能有不同的研究背景和兴趣。老师可以安排小组讨论活动，让学生分享自己的观点和看法，并促进交流和互动。

**（4）推荐更多资源：**人工智能是一个不断发展和变化的领域，学生希望老师能够向他们提供更多的学习资源，例如在线教程、学术研究论文和学术会议等。

## 5、参考资料

[1] GAN 实践：MNIST 手写数字生成 - 吃橘子趁早的文章 - 知乎
https://zhuanlan.zhihu.com/p/366926565

[2] GAN 实战之 Pytorch 使用 CGAN 生成指定 MNIST 手写数字
https://blog.csdn.net/m0_62128864/article/details/123997832

[3] 如何训练 GAN？训练 GAN 的技巧和方法
https://blog.csdn.net/Stephanie2014/article/details/114401268

[4] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. ArXiv, abs/1701.07875.

[5] WGAN(Wasserstein GAN)看这一篇就够啦，WGAN 论文解读
https://blog.csdn.net/m0_62128864/article/details/124258797