大数据分析课程实验报告

1. 程序说明、软硬件环境。
2. 软件环境
3. 编程语言版本：

Python 3.9

Pytorch 2.3.0 + cuda 12.1

1. 编译器：

Python解释器版本为3.9

1. 用到的数据库：

未使用数据库

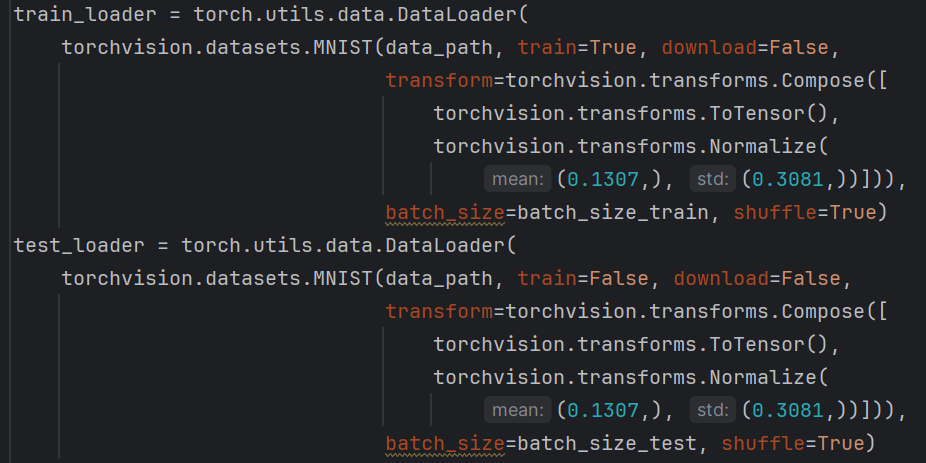
1. 其他：

使用的开发环境为Pycharm + conda

1. 硬件环境
2. CPU：13th Gen Intel(R) Core(TM) i9-13980HX 2.20 GHz
3. GPU：NVIDIA GeForce RTX 4080 Laptop GPU
4. 内存：三星 DDR5 4800MHz 8Bank Group 32 GB (2 × 16 GB) (31.6 GB 可用)
5. 其他：实际硬件环境详见[该网页](https://detail.zol.com.cn/series/16/41373/param_10731201_0_1.html)
6. 数据预处理。
7. 数据预处理方法介绍

数据的处理主要分为两部分，一部分是将读入的数据转化为张量，另一部分是设置数据集的全局平均值和标准差。

1. 数据预处理代码



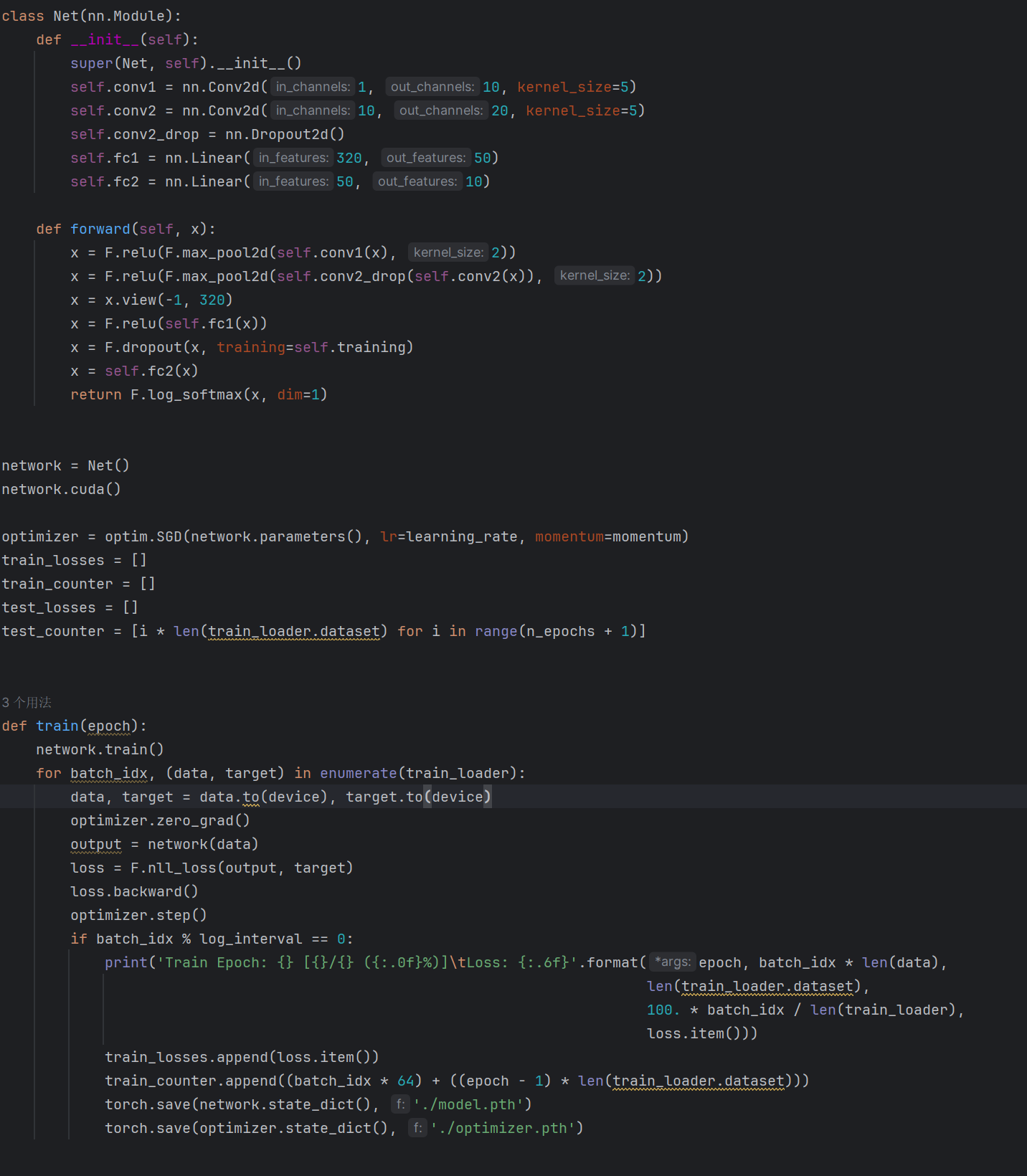
1. 数据预处理代码解释

这里都是使用了torchvision.transforms中已有的函数，即ToTensor()和Normalize()来进行数据预处理的过程。

1. 采用的分类算法介绍、分类算法代码、代码解释（逐行或逐块）。
2. 分类算法介绍

使用的算法为卷积神经网络。具体解释如下：首先神经网络的卷积层会对源数据进行处理，进行特征提取，然后两层全连接层会对处理后的数据进行计算，最后输出预测结果。损失函数会对预测结果的误差进行计算，再计算出损失函数对于整个网络中的每一项参数的偏微分（偏导数）以此获取各个参数要调整的方向的，即实现梯度下降。通过损失函数的结果对网络的反向传播结合计算出的偏微分对每个参数进行调整，以此完成神经网络的一次学习。再选出一部分数据作为测试集作为学习效果的检验，神经网络的学习目标即为最小化测试集的误差和。

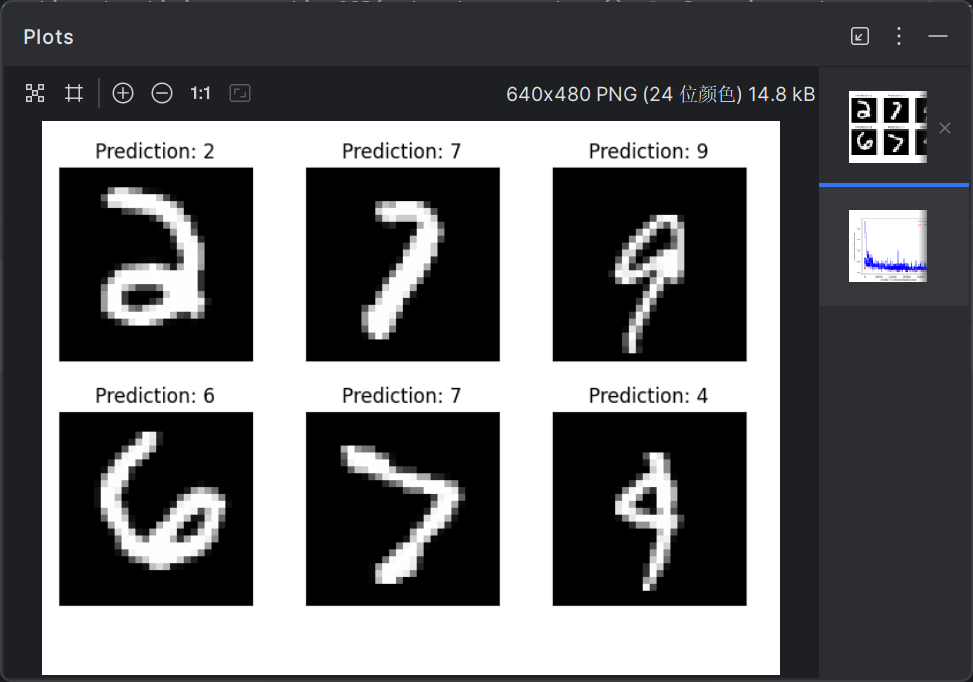
1. 分类算法代码

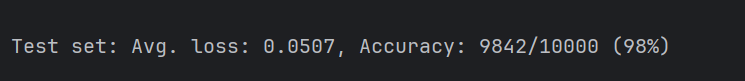


1. 分类算法代码解释

代码使用的大部分都是现有的可以直接套入的函数，这个神经网络组成为先两层卷积层，再两层全连接层；使用null\_loss()作为损失函数，再使用backward()函数进行误差的反向传播，再由定义的optim中的SGD对象optimizer优化器进行参数优化。

1. 实验结果。
2. 结果截图：





1. 实验结果分析：

根据程序的输出，神经网络在进行了10次训练之后预测的准确率已经可以到98%了，而随机挑选的这几个图片的预测结果也是对的，因此当前网络已经有较好的效果了。