# MNIST

目录

[MNIST 1](#_Toc15366)

[实验要求 1](#_Toc28290)

[实验环境 1](#_Toc6517)

[实验准备 2](#_Toc17626)

[实验一：人工神经网络 3](#_Toc9121)

[神经网络结构 3](#_Toc29528)

[实验设置 5](#_Toc21313)

[实验结果 6](#_Toc19658)

[实验二：卷积神经网络 7](#_Toc16660)

[神经网络结构 7](#_Toc31604)

[实验设置 8](#_Toc6344)

[实验结果 9](#_Toc25826)

[实验代码 11](#_Toc14341)

## 实验要求

需要实现的模型：1. 基于Backward Propagation算法的人工神经网络；2. 卷积神经网络。

在MNIST手写字符识别数据集（http://yann.lecun.com/exdb/mnist/）上对实现的两个模型进行实验测试，陈述其原理与结果。

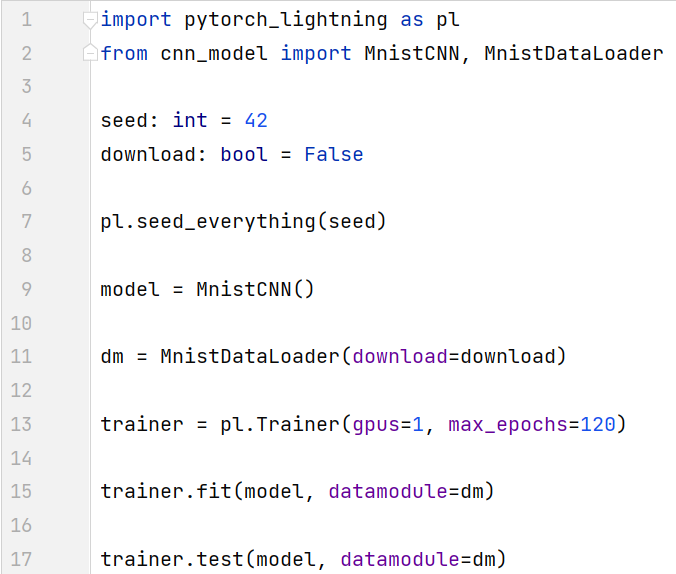
## 实验环境

* Windows 10
* Conda 4.10.3
* Python 3.9
* PyTorch 1.9.1
* PyTorch-Lightning 1.4.9
* CUDA
* DataSpell 2021.3 EAP

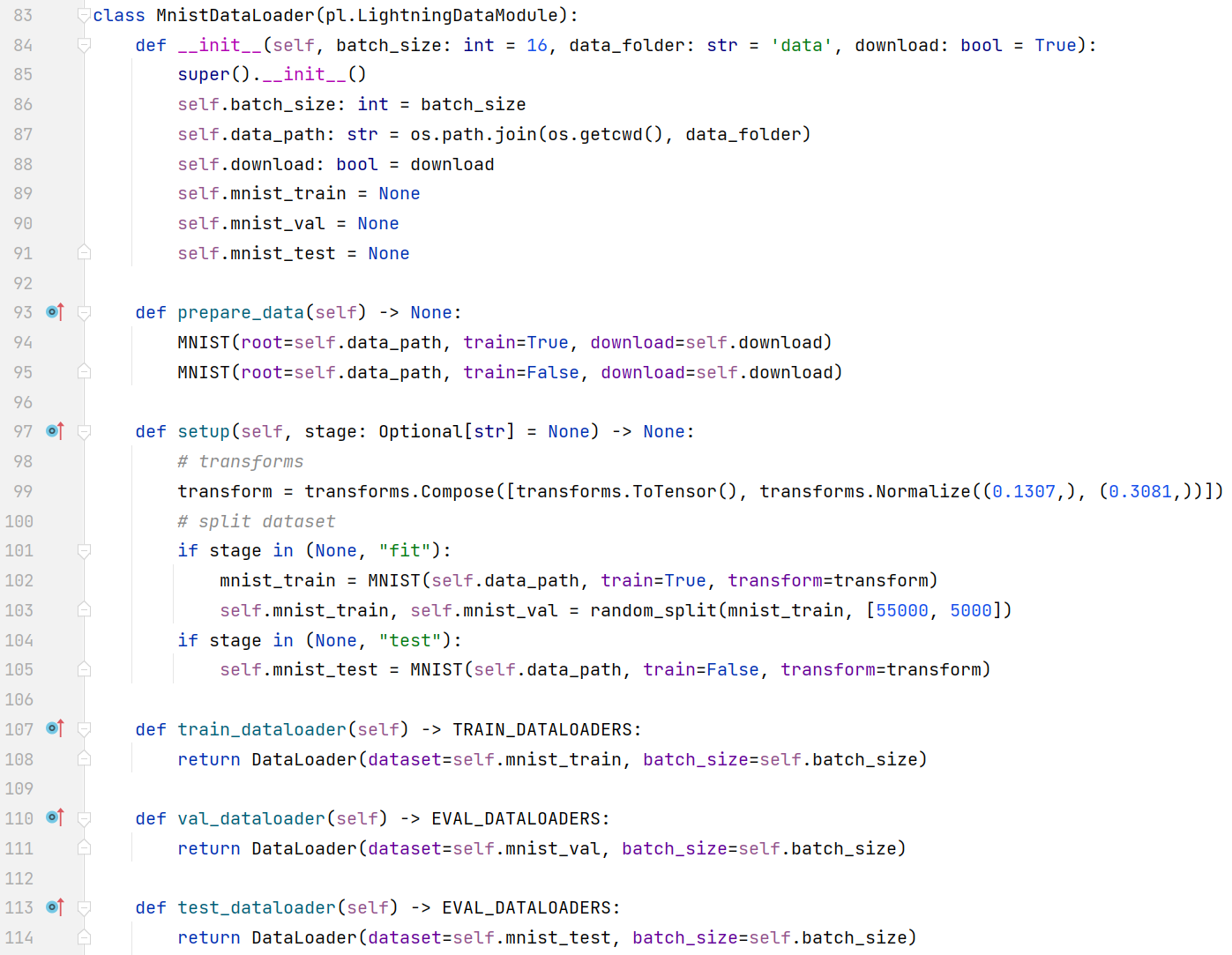
## 实验准备

实验准备阶段，使用PyTorch-Lightning进行训练框架的搭建。

在{model\_name}\_main.py入口脚本（例如cnn\_main.py）中设置Global Seed为42，使用自定义的MnistDataLoader作为训练数据，使用pl.Trainer()对模型进行训练，自定义是否使用GPU、epoch数等参数：



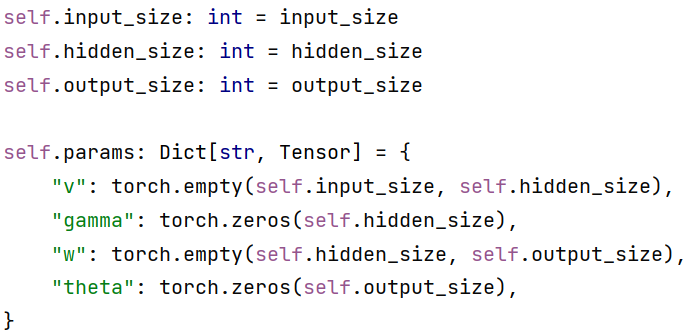
在{model\_name}\_model.py模型脚本（例如cnn\_model.py）中定义MnistDataLoader，用于下载MNIST数据集，并将其进行向量化、正则化，划分9:1的训练-验证集，并指定batch\_size参数：



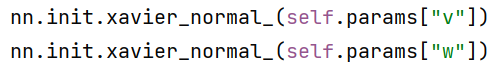
## 实验一：人工神经网络

### 神经网络结构

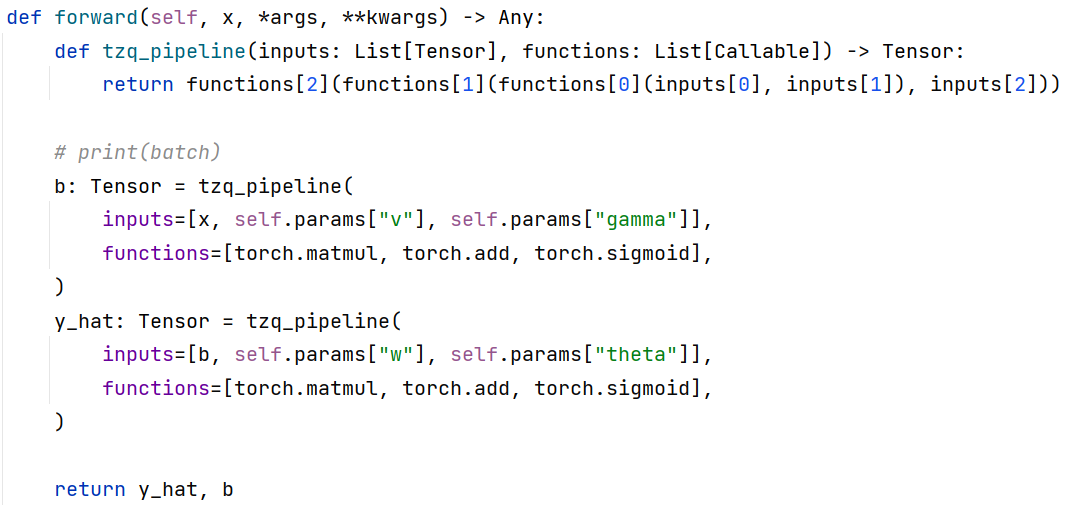
本实验中实现的人工神经网络由**两个全连接层**组成，拥有**两个权重矩阵（weight matrix）**与**两个偏置向量（bias vector）**，每个全连接层后使用Sigmoid作为激活函数：



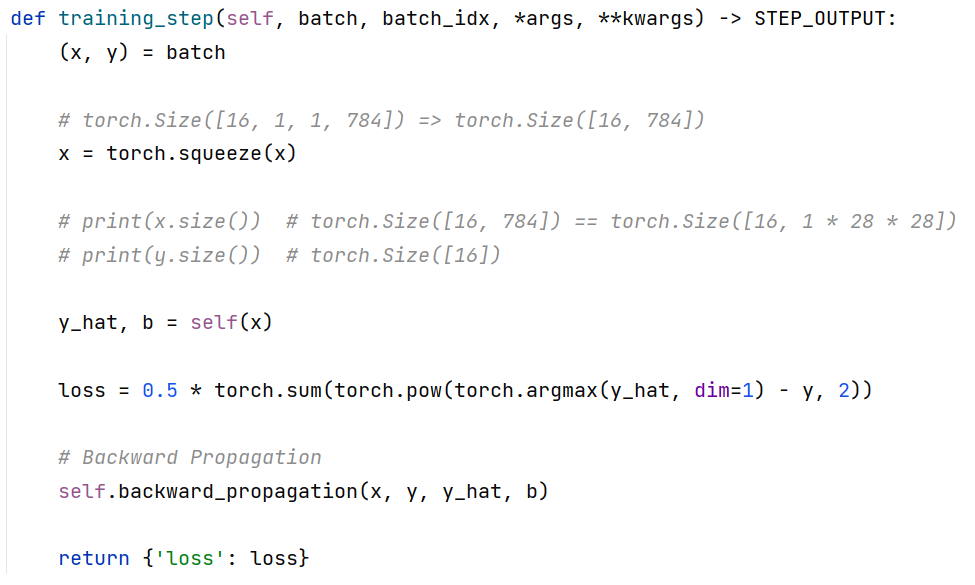
本实验对两个权重矩阵进行**Xavier初始化**：



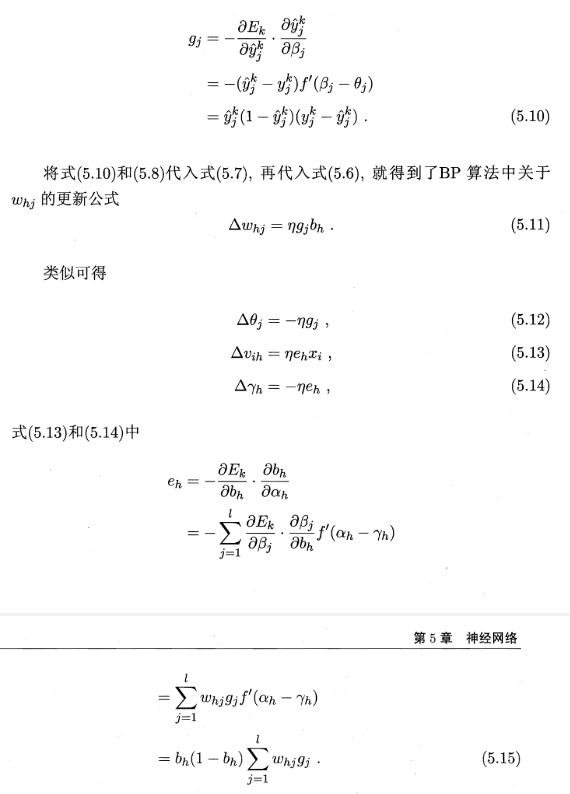
在前向传播函数中，构建tzq\_pipeline函数进行矩阵数据的运算，并返回预测结果y\_hat与中间层结果b，其中，当计算出b与y\_hat时，对两者使用Sigmoid函数进行激活：

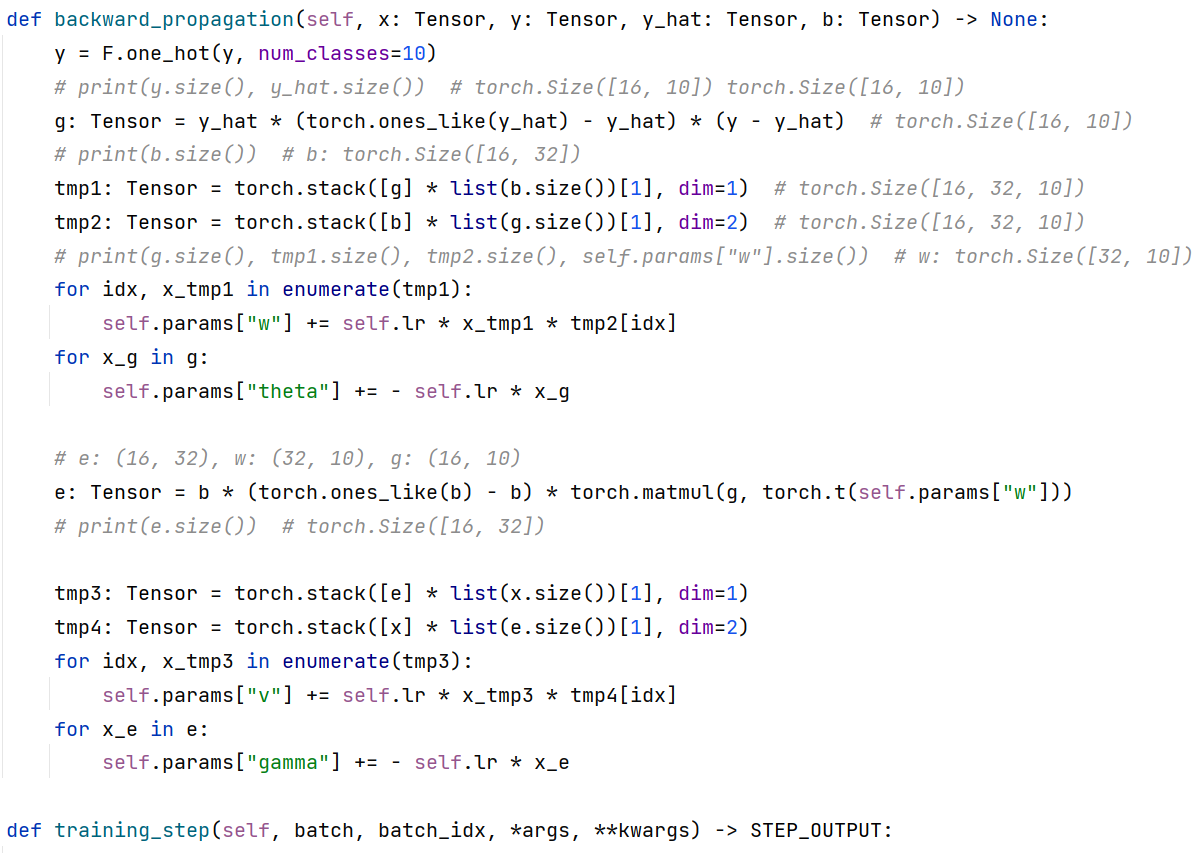


在每次训练步（training\_step）中，对一个batch的数据进行预测，使用**均方误差**进行loss计算，并对网络参数进行**反向传播**（backward propagation）：



在反向传播过程中，模型对均方误差进行求导，**沿着梯度下降的方向进行权重更新**，以达到凸函数极点；模型根据《机器学习（周志华）》5.3 误差逆传播算法中的权重更新方程对权重进行更新：



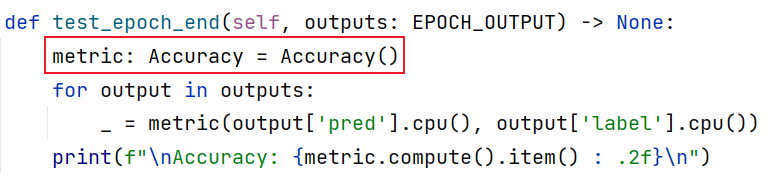


### 实验设置

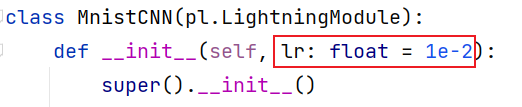
模型使用**均方误差**（Mean Square Error）作为**损失函数**（Loss Function）。



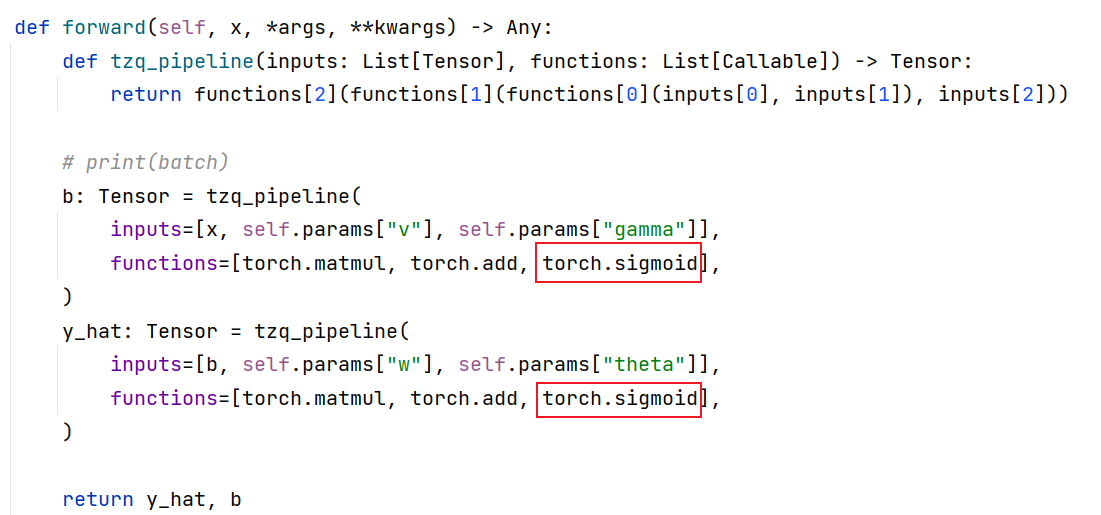
模型使用**精度**（Accuracy）作为**测试标准**。



模型设置**学习率**为**1e-2**。



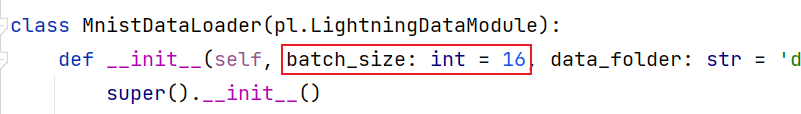
模型使用**Sigmoid**函数作为**激活函数**：



模型设置**epoch**次数为**64**。

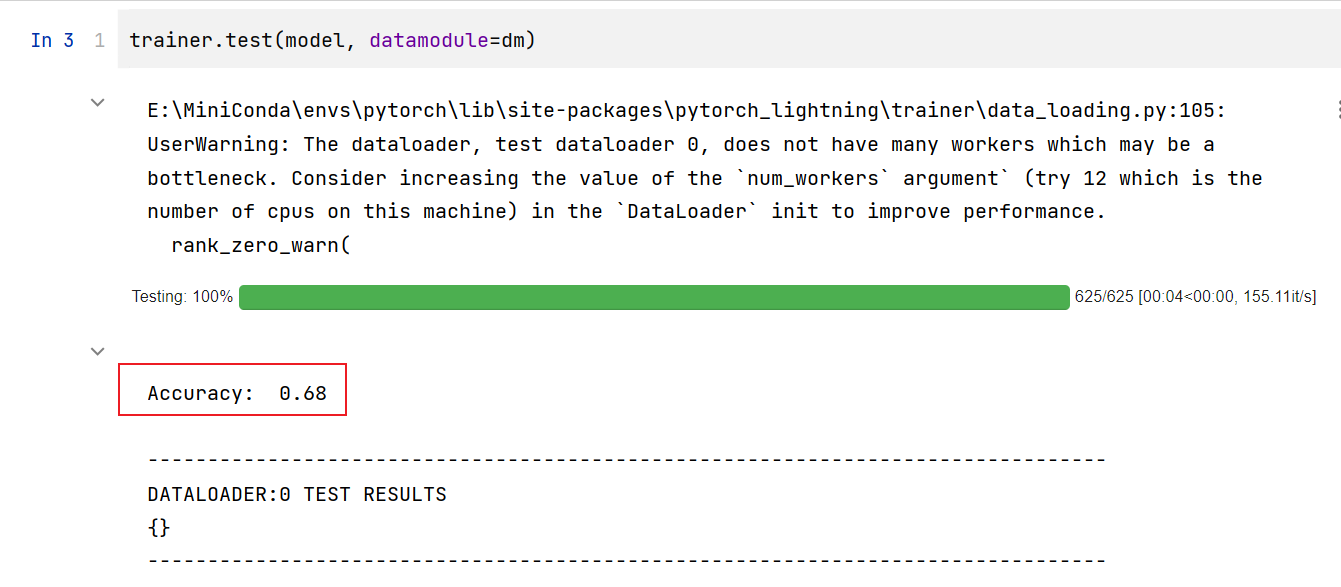


模型设置**batch\_size**为**16**。



### 实验结果

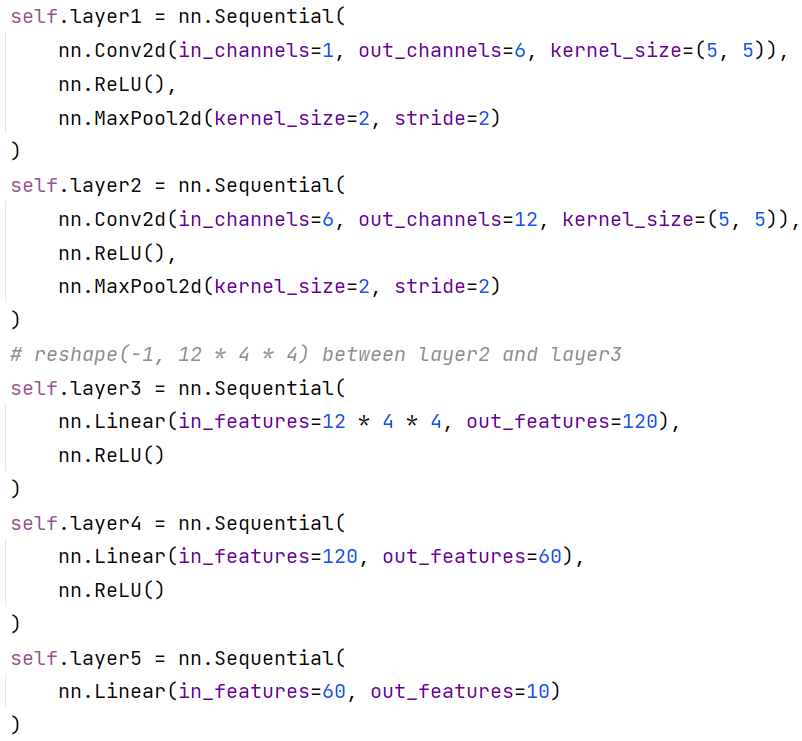
在经过充分的训练后，使用测试数据集对模型进行测试，最终得到**68%**的**预测正确率**，达到了较高的预测精度。



## 实验二：卷积神经网络

### 神经网络结构

本实验设计的CNN网络由五层网络模块组成，其中第一、二个模块运算卷积操作，第三、四个模块的带有ReLU操作的线性层，第五个模块是输出结果的线性层：



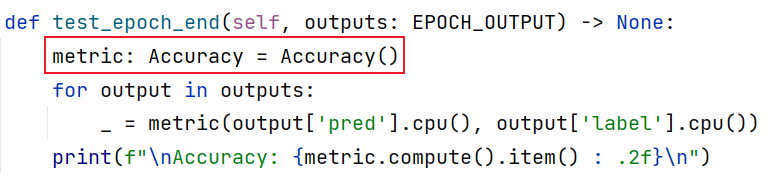
模型使用5×5的卷积核对图像进行卷积，在卷积之后使用ReLU作为激活函数（Activation Function） ，并使用size为2×2、stride为2的池化核对进行最大池化操作。

### 实验设置

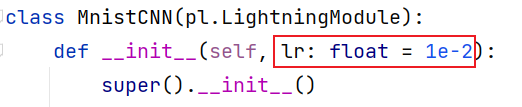
模型使用**交叉熵**（Cross Entropy）作为**损失函数**（Loss Function）。



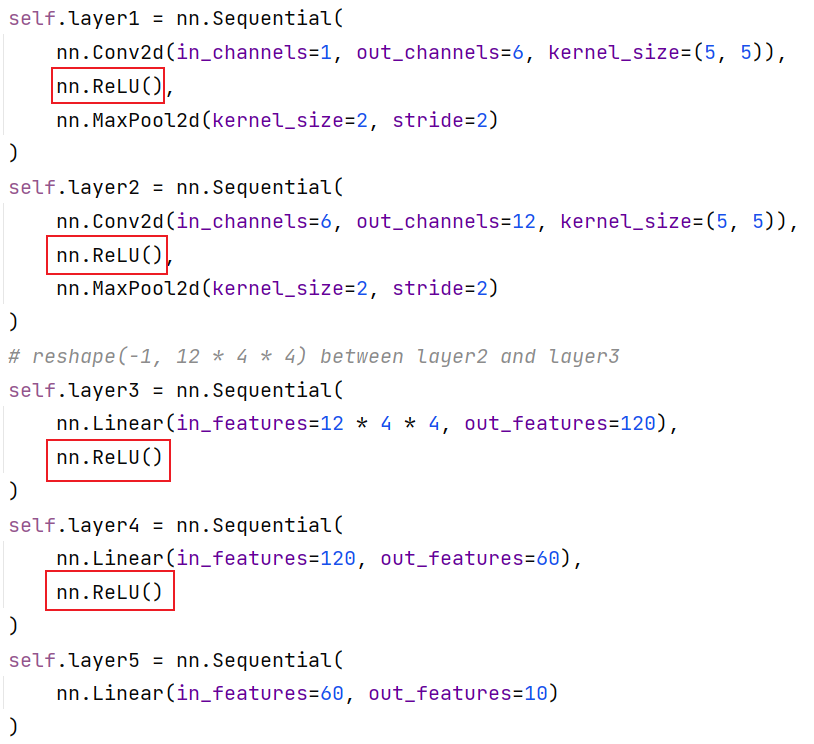
模型使用**精度**（Accuracy）作为**测试标准**。



模型设置**学习率**为**1e-2**。



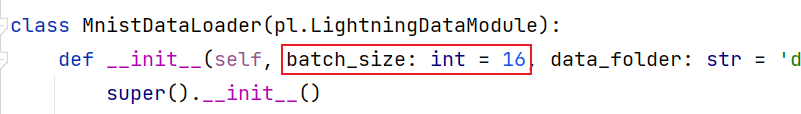
模型使用**ReLU**函数作为**激活函数**，因为相对于Sigmoid与Tanh，ReLU函数具有以下优势：1. 在误差进行反向传播时，可以**缓解梯度消失**；2. ReLU会使一部分神经元的输出为0，造成了网络的稀疏性，并且减少了参数的相互依存关系，**缓解了过拟合问题的发生**；3. 相对于Sigmoid与Tanh，ReLU函数**求导简单，计算量较小**，节约本就不多的计算资源。



模型设置**epoch**次数为**64**。



模型设置**batch\_size**为**16**。

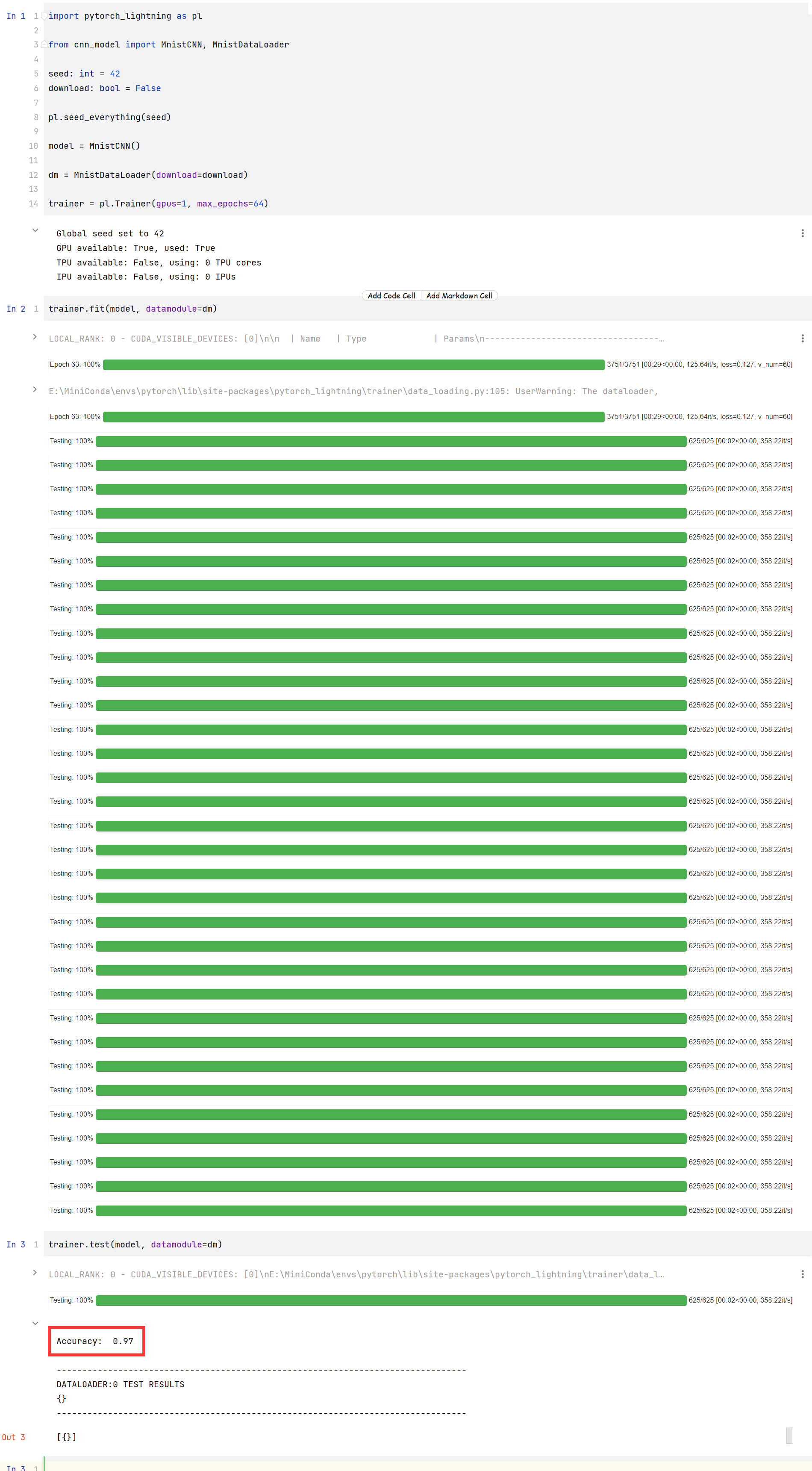


模型使用**GPU**进行训练加速。



### 实验结果

在经过充分的训练后，使用测试数据集对模型进行测试，最终得到**97%**的**预测正确率**，达到了极高的预测精度。



## 实验代码

实验的代码在提交的压缩包中的“代码”文件夹下的“ANNwithPyTorch”文件夹下，其中：

1. data文件夹中包含了预下载的MNIST数据集；
2. ann\_model.py中包含了ANN模型代码与MNIST数据集处理代码，ann\_main.py是ANN实验的入口脚本；
3. cnn\_model.py中包含了CNN模型代码与MNIST数据集处理代码，cnn\_main.py是CNN实验的入口脚本。

