# 手写数字识别

## 1 概述

任务：使用卷积神经网络完成手写数字识别任务。

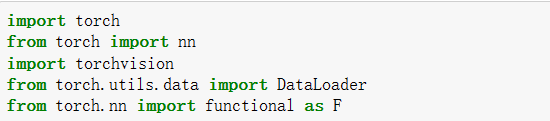
数据集：MNIST。MNIST数据集由60000张训练集图片与10000张测试集图片构成，图片为28\*28的灰度图像。

解决方案：使用torchvision加载MINST数据集，继承torch.nn.Module类搭建网络，损失函数使用交叉熵损失，优化器使用Adam。

2 解决方案

2.1 导入相关库：

实验使用pytorch框架，torchvision用于下载与加载数据集。



2.2 获取数据集

首先创建对图片进行预处理的transform，将图片转换为tensor并进行归一化处理，这里按照均值为0.5，方差为0.5进行处理。

其次加载训练集与测试集，使用torchvision.datasets.MNIST即使在数据集，root代指数据集的路径，train=Train表示这是训练集，transform表示对数据集进行transform处理，download=True表示若数据集不存在则下载。

最后创建DataLoader对象，用于训练与测试，训练集对象与测试集对象分别传入训练集与测试集，batch\_size代指批处理大小，shuffle=True表示对数据随机打乱。



2.3 定义模型

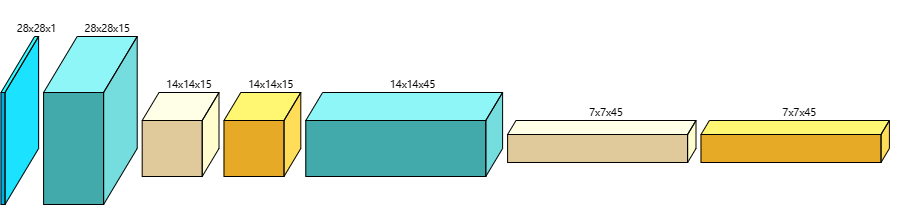
模型继承torch.nn.Module,需要调用父类的初始化函数。

这里创建了两个卷积层，两个池化层，一个全连接层。卷积层的5个参数分别为输入通道，输出通道，卷积核大小，步长，填充。填充的作用是控制卷积后图形的大小。池化层选择的是最大池化。全连接层的两个参数分别是将卷积输出图像展平后的节点数乘输出通道数与分类的类别数。

模型运算流程：输入x为(batch\_size,1,28,28),经第一层卷积后变为(batch\_size,10,28,28),经第一层池化变为(batch\_size,10,14,14),使用relu函数对其激活,经第二层卷积后变为((batch\_size,30,14,14)),经第二层池化变为(batch\_size,30,7,7),使用relu函数对其激活，数据展平，其形状变为(batch\_size,7\*7\*30),经全连接层，其形状变为(batch\_size,10)。



卷积-池化 结构

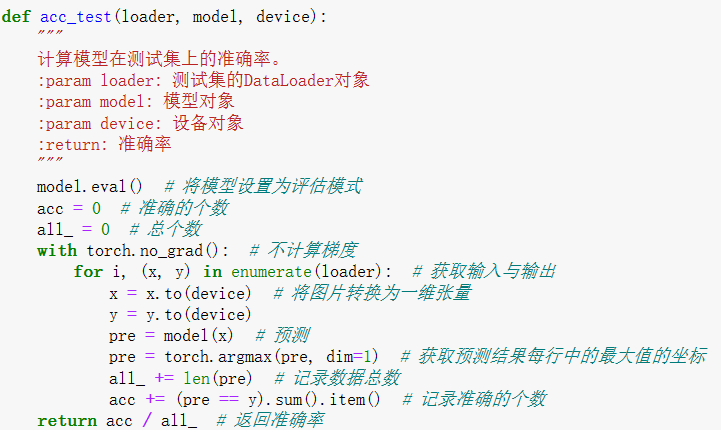


2.4 定义评估函数

评估函数需要传入待评估的dataloader对象，模型与训练设备。训练设备需要保证全局数据在同一设备上。

首先将模型转变为评估模式，定义两个变量分别存储准确的个数与总个数。使用torch.no\_grad()不计算梯度，对评估数据集进行遍历，将输入数据与标签均迁移到训练设备上后进行模型预测，模型的输出为(batch\_size,10)，取出第一维的最大值的下标即为预测值。

记录准确的个数与参与评估的个数，其比值即为准确率



2.5 定义训练函数

损失函数使用交叉熵损失，优化器使用Adam，batch\_size大小设置为128，训练5个epoch，学习率0.01。

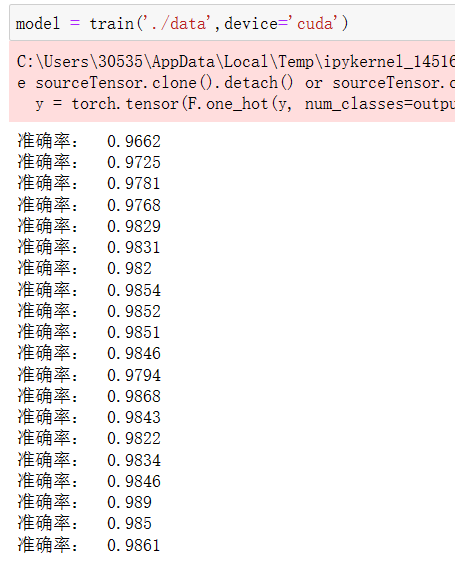
首先创建损失函数，模型，优化器，获取训练集与测试集的Dataloader，将模型迁移到对应设备上开始训练。

外层循环指对数据训练epochs次，内层循环对Dataloader进行处理与训练。首先读取出每个batch的训练数据与标签，将标签使用one\_hot函数转变为(batch\_size,10),并将数据迁移到相应设备上。由于pytorch的优化器默认会梯度累加，因此每次循环需要先将梯度归0，之后，执行预测，计算损失，反向传播，梯度更新。每100个batch对测试集进行一次评估。



2.6 训练

这里只传入数据路径与训练设备，其它均使用train函数的默认参数。



3 总结

训练参数如下：优化器：Adam

损失函数：交叉熵

学习率：0.01

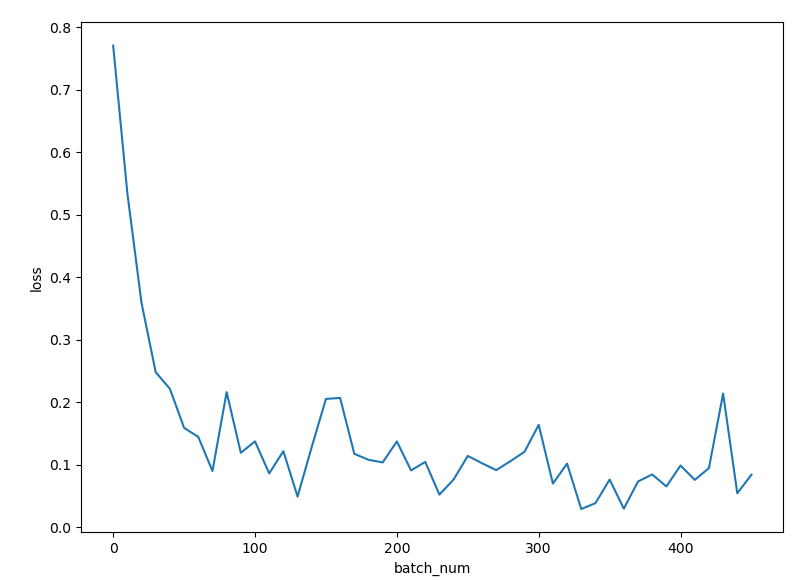
batch\_size:128

epochs:5

训练设备:GPU

在以上参数不变的情况下，增加卷积核的数量，在一定程度上可以加快模型收敛速度，但最终的准确率基本稳定在98%。

Loss值下降曲线



准确率上升曲线：

