# 计算机视觉实验报告（第二次作业）

## 实验目的

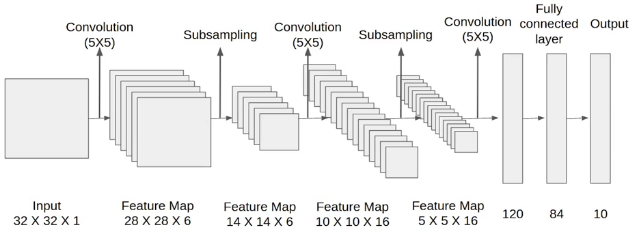
搭建一个pytorch框架，并且利用该框架选用MNIST手写数字数据集在LeNet网络结构下训练，获得一个有较好识别数字性能的模型。

## 实现说明

1. **MNIST数据集介绍：**

* MNIST 数据集是一个广泛用于机器学习和深度学习领域的经典数据集之一。它包含了手写数字的灰度图像，每个图像都是 28x28 像素大小。MNIST 数据集共有 70000 张图像，其中包括 60000 张用作训练样本，10000 张用作测试样本。
* 这些手写数字图像的标签是它们所代表的数字，范围从 0 到 9。因此，MNIST 数据集可以用于图像分类任务，即通过训练模型来识别图像中所包含的数字。
* MNIST 数据集因其简单和易于使用而成为机器学习领域的基准数据集之一。它被广泛用于验证新的机器学习算法的有效性和性能，并且为研究者提供了一个标准的测试平台，用于比较不同方法的性能和效果。

1. **LeNe-5介绍：**



* 卷积层1：使用5×5的卷积核在输入图像上进行卷积操作，得到28×28×6的特征图1。这意味着我们使用6个5×5的卷积核对单通道的输入图像进行卷积，输出为6个通道，每个通道的大小为28×28。
* 下采样(池化)：使用一个2×2的avgPooling(平均池化)对特征图1进行下采样，得到14×14×6的特征图2。这个操作有助于减少特征图的尺寸，同时保留重要信息。
* 卷积层2：使用5×5的卷积核在特征图2上进行卷积，得到10×10×16的特征图3。这次我们使用16个5×5的卷积核，在之前的特征图上进行卷积操作，输出16个通道的特征图，每个通道大小为10×10。
* 下采样(池化)：使用一个2×2的avgPooling(平均池化)对特征图3进行下采样，得到5×5×16的特征图4。同样地，这个操作有助于减少特征图的尺寸，减少计算量。
* 卷积层3：使用120个5×5的卷积核在特征图4上卷积，得到一个1×1×120的向量。这是一个全连接层之前的最后一个卷积层。它将所有16个通道的5×5的特征图转换为一个长度为120的向量。
* 全连接层1：将卷积层3的输出连接到一个具有120个神经元的全连接层，然后再连接到一个具有84个神经元的全连接层。
* 全连接层2：将全连接层1的输出连接到一个具有10个神经元的全连接层，其中每个神经元对应于数据集中的一个类别（0到9）。这个全连接层产生最终的输出，用于预测输入图像属于哪个数字类别。

1. 实验整体流程介绍：

* 使用Pytorch搭建网络结构
* 载入MNIST数据集

由于MNIST数据集的原始大小为28\*28，而LeNet5的输入尺寸为32\*32，因此在读取数据的时候要先做一步resize处理，将图片缩放，这里使用pytorch中的transforms来转换格式。

同时，由于raw数据差异较大并且数值通常都很大，因此tanh求导得到的导数可能接近于0，可能造成梯度丢失。为了解决这一问题，最好使用既有的均值和方差，将每个数据样本进行标准化，使得数据最后呈现均值为0方差为1的分布。

1. 下载数据集

train\_dataset = datasets.MNIST(root='data',

train=True,

transform=transforms,

download=True)

valid\_dataset = datasets.MNIST(root='data',

train=False,

transform=transforms)

1. 根据指定数据集和batch-size形成数据流（数据加载器）

train\_loader = DataLoader(dataset=train\_dataset,

batch\_size=BATCH\_SIZE,

shuffle=True)

valid\_loader = DataLoader(dataset=valid\_dataset,

batch\_size=BATCH\_SIZE,

shuffle=False)

1. 设置学习步长、优化器和损失函数开始训练

LEARNING\_RATE=0.001

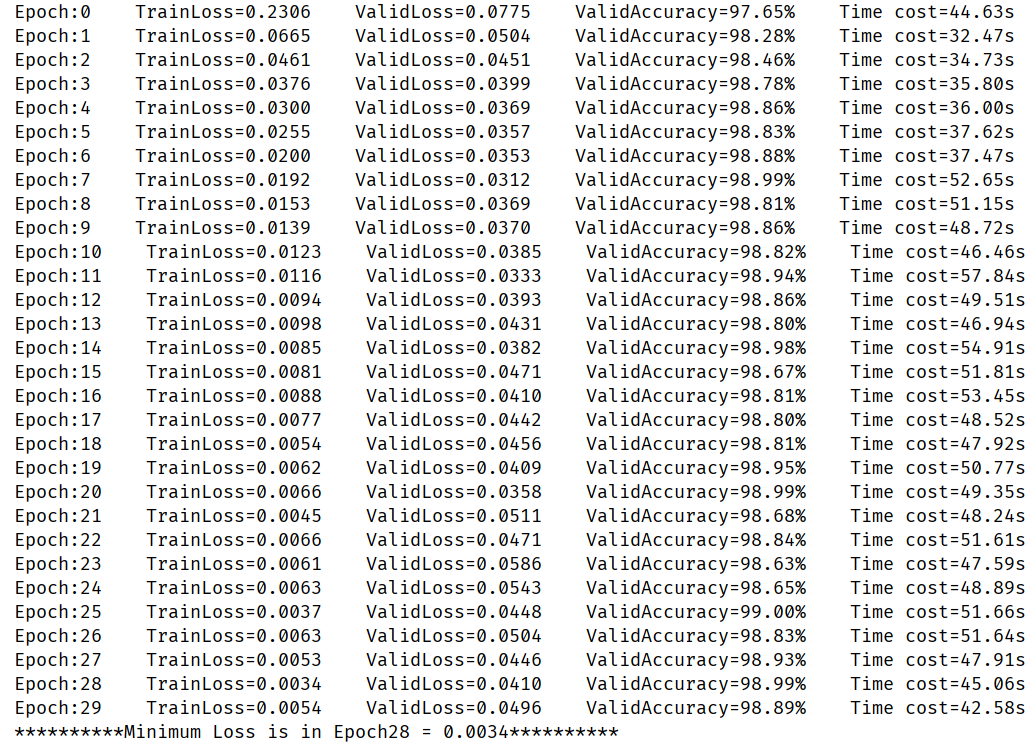
optimizer = torch.optim.Adam(model.parameters(), lr=LEARNING\_RATE)

criterion = nn.CrossEntropyLoss()

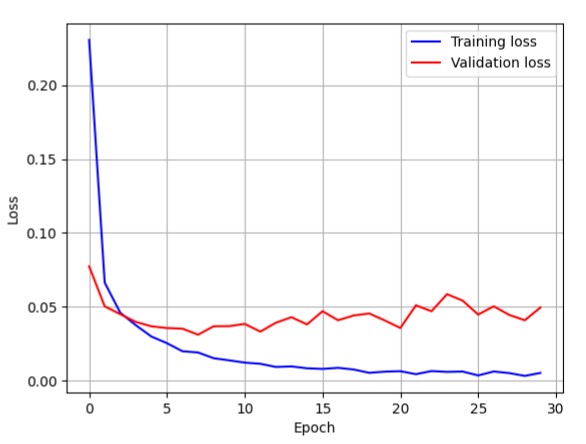
1. 对训练结果进行验证评估操作，并且进行适当的可视化操作

## 结果截图

1. 终端输出的整个训练和验证过程日志信息



1. 对训练和验证结果进行可视化操作



* 可以看出，对于30个Epoch，验证损失已经接近于稳定，并开始逐渐呈现上升的态势，因此30个Epoch对于这个模型来说已经足够了。

## 运行说明

在训练LeNet-5网络时，首先需要读取MNIST，如果是首次加载该数据集，他会前往官网下载MNIST并将其保存在data目录下，下载过一次该数据集后，以后如果再次训练该网络，可以不用再次下载。再定义好了训练的参数（学习率、训练轮次等）、优化器、损失函数后开始训练模型，等待训练结束后，计算训练和验证的准确率和损失，再控制台输出，并将其可视化后，根据训练时间的不同唯一保存在result目录下，全部训练结束后，将训练获得的模型保存在Model目录下。

## 实验小结

学习使用了pytorch的基本操作，学会了使用pytorch搭建一个经典的网络框架，加深了对深度学习神经网络中一些术语的理解，锻炼了代码能力。同时对LeNet5网络框架结构有了一个很好的了解，了解了一张图片从开始到进入网络在到最后分类输出结果整体的过程有了一个更深的了解，这次试验让人受益匪浅。