机器学习第三次实验实验报告

# 实验要求

在这个练习中，需要用Python实现LeNet5来完成对MNIST数据集中 0-9 10个手写数字的分类。代码只能使用python实现，不能使用PyTorch或TensorFlow框架。

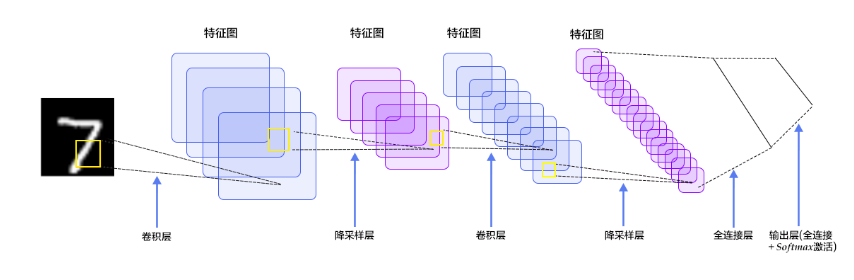
# 实验环境

Python3.10

Vs Code

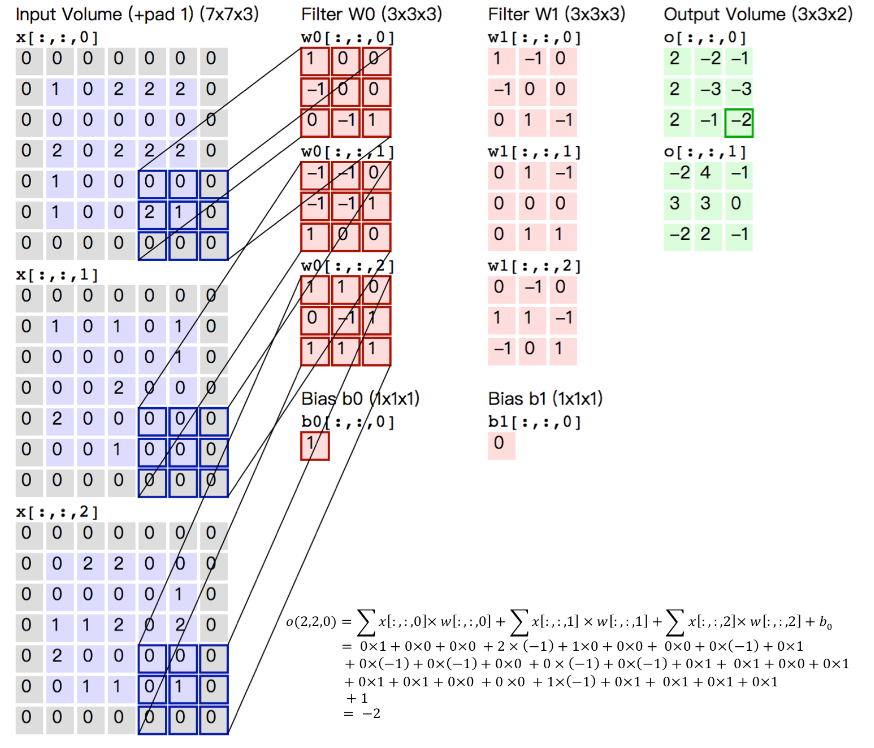
# 实验准备

LeNet-5是一个较简单的卷积神经网络。下图显示了其结构：输入的二维图像，先经过两次卷积层到池化层，再经过全连接层，最后使用softmax分类作为输出层。



## 卷积层

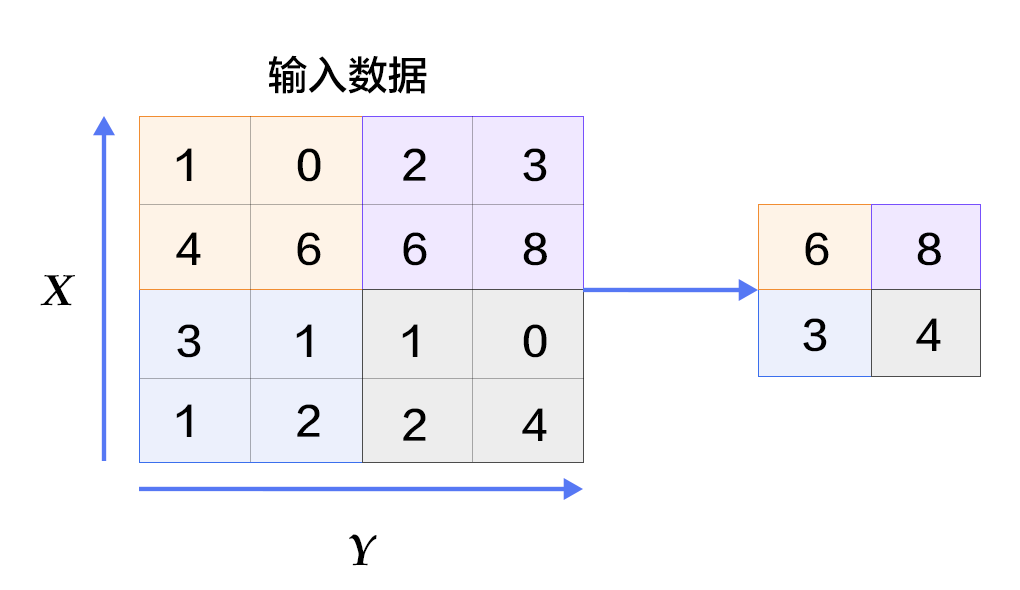
卷积层是卷积神经网络的核心基石。在图像识别里我们提到的卷积是二维卷积，即离散二维滤波器（也称作卷积核）与二维图像做卷积操作，简单的讲是二维滤波器滑动到二维图像上所有位置，并在每个位置上与该像素点及其领域像素点做内积。卷积操作被广泛应用与图像处理领域，不同卷积核可以提取不同的特征，例如边沿、线性、角等特征。在深层卷积神经网络中，通过卷积操作可以提取出图像低级到复杂的特征。



计算过程如上图示例。输入图像大小为H=5，W=5，D=3，这个示例图中包含两（用K表示）组卷积核，即图中滤波器W0和W1。在卷积计算中，通常对不同的输入通道采用不同的卷积核，如图示例中每组卷积核包含（D=3）个3×3（用F×F表示）大小的卷积核。另外，这个示例中卷积核在图像的水平方向（W方向）和垂直方向（H方向）的滑动步长为2（用S表示）；对输入图像周围各填充1（用P表示）个0，即图中输入层原始数据为蓝色部分，灰色部分是进行了大小为1的扩展，用0来进行扩展。经过卷积操作得到输出为3×3×2（用Ho×Wo×K表示）大小的特征图，即3×3大小的2通道特征图，其中Ho计算公式为：Ho=(H−F+2×P)/S+1，Wo同理。 而输出特征图中的每个像素，是每组滤波器与输入图像每个特征图的内积再求和，再加上偏置bo，偏置通常对于每个输出特征图是共享的。输出特征图o[:,:,0]中的最后一个−2计算如上图右下角公式所示。

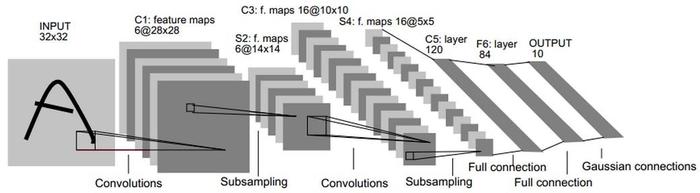
在卷积操作中卷积核是可学习的参数，经过上面示例介绍，每层卷积的参数大小为D×F×F×K。卷积层的参数较少，这也是由卷积层的主要特性即局部连接和共享权重所决定。

## 池化层



池化是非线性下采样的一种形式，主要作用是通过减少网络的参数来减小计算量，并且能够在一定程度上控制过拟合。通常在卷积层的后面会加上一个池化层。池化包括最大池化、平均池化等。其中最大池化是用不重叠的矩形框将输入层分成不同的区域，对于每个矩形框的数取最大值作为输出层，如上图所示。

## Lenet-5



LeNet-5共有7层，不包含输入，每层都包含可训练参数；每个层有多个Feature Map，每个FeatureMap通过一种卷积滤波器提取输入的一种特征，然后每个FeatureMap有多个神经元。

### INPUT层-输入层

首先是数据 INPUT 层，输入图像的尺寸统一归一化为32\*32。

### C1层-卷积层

输入图片：32\*32

卷积核大小：5\*5

卷积核种类：6

输出featuremap大小：28\*28 （32-5+1）=28

神经元数量：28\*28\*6

可训练参数：（5\*5+1) \* 6（每个滤波器5\*5=25个unit参数和一个bias参数，一共6个滤波器）

连接数：（5\*5+1）\*6\*28\*28=122304

详细说明：对输入图像进行第一次卷积运算（使用 6 个大小为 5\*5 的卷积核），得到6个C1特征图（6个大小为28\*28的 feature maps, 32-5+1=28）。我们再来看看需要多少个参数，卷积核的大小为5\*5，总共就有6\*（5\*5+1）=156个参数，其中+1是表示一个核有一个bias。对于卷积层C1，C1内的每个像素都与输入图像中的5\*5个像素和1个bias有连接，所以总共有156\*28\*28=122304个连接（connection）。有122304个连接，但是我们只需要学习156个参数，主要是通过权值共享实现的。

### S2层-池化层（下采样层）

输入：28\*28

采样区域：2\*2

采样方式：4个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过sigmoid

采样种类：6

输出featureMap大小：14\*14（28/2）

神经元数量：14\*14\*6

连接数：（2\*2+1）\*6\*14\*14

S2中每个特征图的大小是C1中特征图大小的1/4。

详细说明：第一次卷积之后紧接着就是池化运算，使用 2\*2核 进行池化，于是得到了S2，6个14\*14的 特征图（28/2=14）。S2这个pooling层是对C1中的2\*2区域内的像素求和乘以一个权值系数再加上一个偏置，然后将这个结果再做一次映射。同时有5x14x14x6=5880个连接。

### C3层-卷积层

输入：S2中所有6个或者几个特征map组合

卷积核大小：5\*5

卷积核种类：16

输出featureMap大小：10\*10 (14-5+1)=10

C3中的每个特征map是连接到S2中的所有6个或者几个特征map的，表示本层的特征map是上一层提取到的特征map的不同组合

存在的一个方式是：C3的前6个特征图以S2中3个相邻的特征图子集为输入。接下来6个特征图以S2中4个相邻特征图子集为输入。然后的3个以不相邻的4个特征图子集为输入。最后一个将S2中所有特征图为输入。

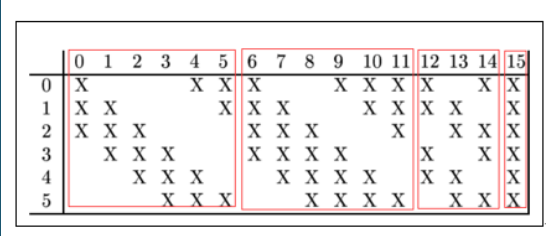
则：可训练参数：6\*(3\*5\*5+1)+6\*(4\*5\*5+1)+3\*(4\*5\*5+1)+1\*(6\*5\*5+1)

=1516

连接数：10\*10\*1516=151600

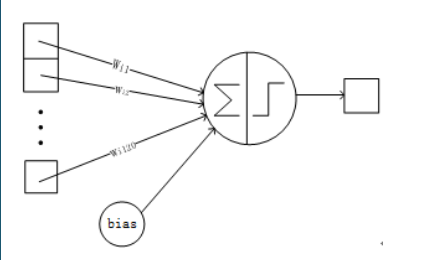
详细说明：

第一次池化之后是第二次卷积，第二次卷积的输出是C3，16个10x10的特征图，卷积核大小是 5\*5. 我们知道S2 有6个 14\*14 的特征图，怎么从6 个特征图得到 16个特征图了？这里是通过对S2 的特征图特殊组合计算得到的16个特征图。具体如下：

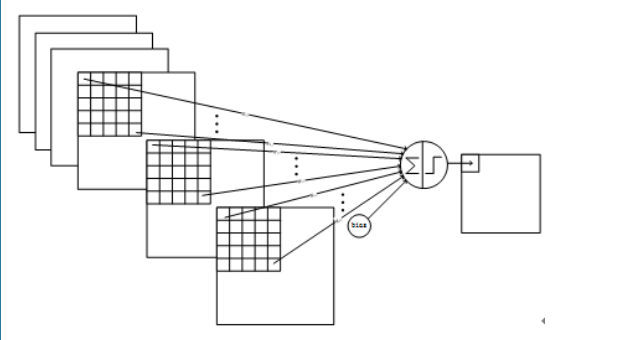


C3的前6个feature map（对应上图第一个红框的6列）与S2层相连的3个feature map相连接（上图第一个红框），后面6个feature map与S2层相连的4个feature map相连接（上图第二个红框），后面3个feature map与S2层部分不相连的4个feature map相连接，最后一个与S2层的所有feature map相连。卷积核大小依然为5\*5，所以总共有6\*(3\*5\*5+1)+6\*(4\*5\*5+1)+3\*(4\*5\*5+1)+1\*(6\*5\*5+1)=

1516个参数。而图像大小为10\*10，所以共有151600个连接。



C3与S2中前3个图相连的卷积结构如下图所示：



上图对应的参数为 3\*5\*5+1，一共进行6次卷积得到6个特征图，所以有6\*（3\*5\*5+1）参数。 为什么采用上述这样的组合了？论文中说有两个原因：1）减少参数，2）这种不对称的组合连接的方式有利于提取多种组合特征。

### S4层-池化层（下采样层）

输入：10\*10

采样区域：2\*2

采样方式：4个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过sigmoid

采样种类：16

输出featureMap大小：5\*5（10/2）

神经元数量：5\*5\*16=400

连接数：16\*（2\*2+1）\*5\*5=2000

S4中每个特征图的大小是C3中特征图大小的1/4

详细说明：S4是pooling层，窗口大小仍然是2\*2，共计16个feature map，C3层的16个10x10的图分别进行以2x2为单位的池化得到16个5x5的特征图。有5x5x5x16=2000个连接。连接的方式与S2层类似。

### C5层-卷积层

输入：S4层的全部16个单元特征map（与s4全相连）

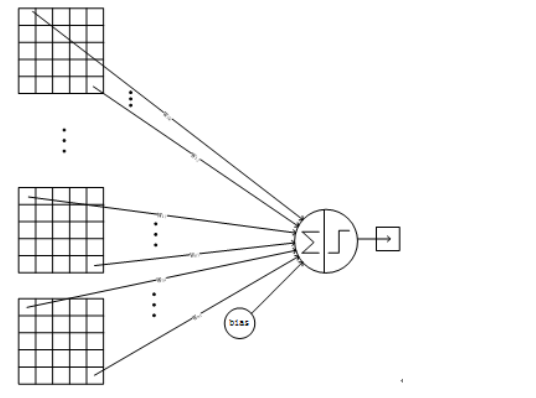
卷积核大小：5\*5

卷积核种类：120

输出featureMap大小：1\*1（5-5+1）

可训练参数/连接：120\*（16\*5\*5+1）=48120

详细说明：C5层是一个卷积层。由于S4层的16个图的大小为5x5，与卷积核的大小相同，所以卷积后形成的图的大小为1x1。这里形成120个卷积结果。每个都与上一层的16个图相连。所以共有(5x5x16+1)x120 = 48120个参数，同样有48120个连接。C5层的网络结构如下：



### F6层-全连接层

输入：c5 120维向量

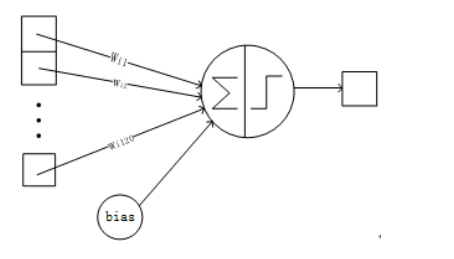
计算方式：计算输入向量和权重向量之间的点积，再加上一个偏置，结果通过sigmoid函数输出。

可训练参数:84\*(120+1)=10164

详细说明：6层是全连接层。F6层有84个节点，对应于一个7x12的比特图，-1表示白色，1表示黑色，这样每个符号的比特图的黑白色就对应于一个编码。该层的训练参数和连接数是(120 + 1)x84=10164。ASCII编码图如下：



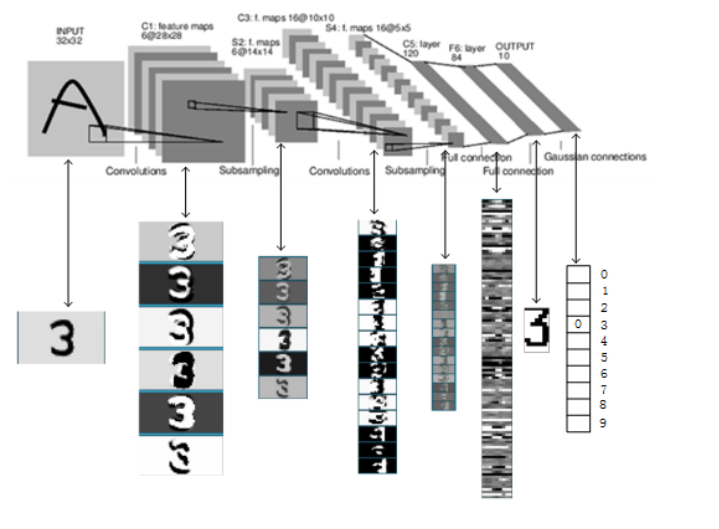
F6层的连接方式如下：



### Output层-全连接层

Output层也是全连接层，共有10个节点，分别代表数字0到9，且如果节点i的值为0，则网络识别的结果是数字i。采用的是径向基函数（RBF）的网络连接方式。假设x是上一层的输入，y是RBF的输出，则RBF输出的计算方式是：

上式w\_ij 的值由i的比特图编码确定，i从0到9，j取值从0到7\*12-1。RBF输出的值越接近于0，则越接近于i，即越接近于i的ASCII编码图，表示当前网络输入的识别结果是字符i。该层有84x10=840个参数和连接。



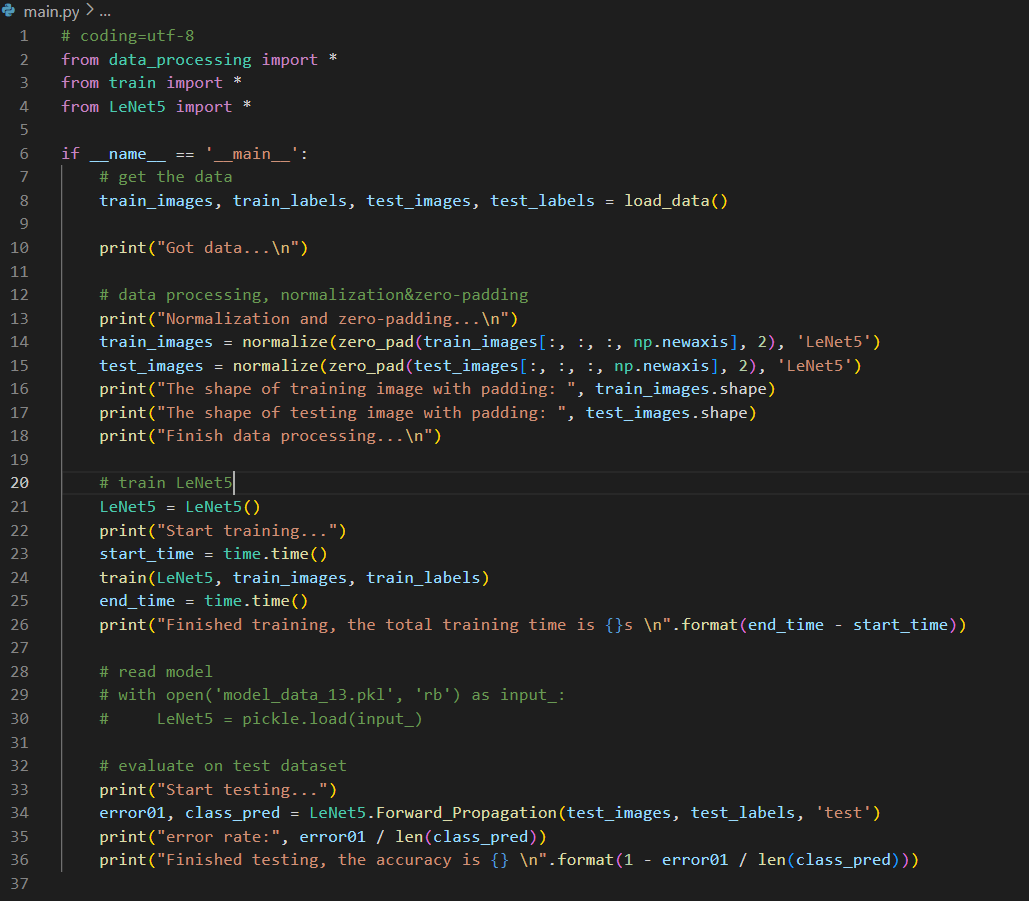
# 实验内容

## Main

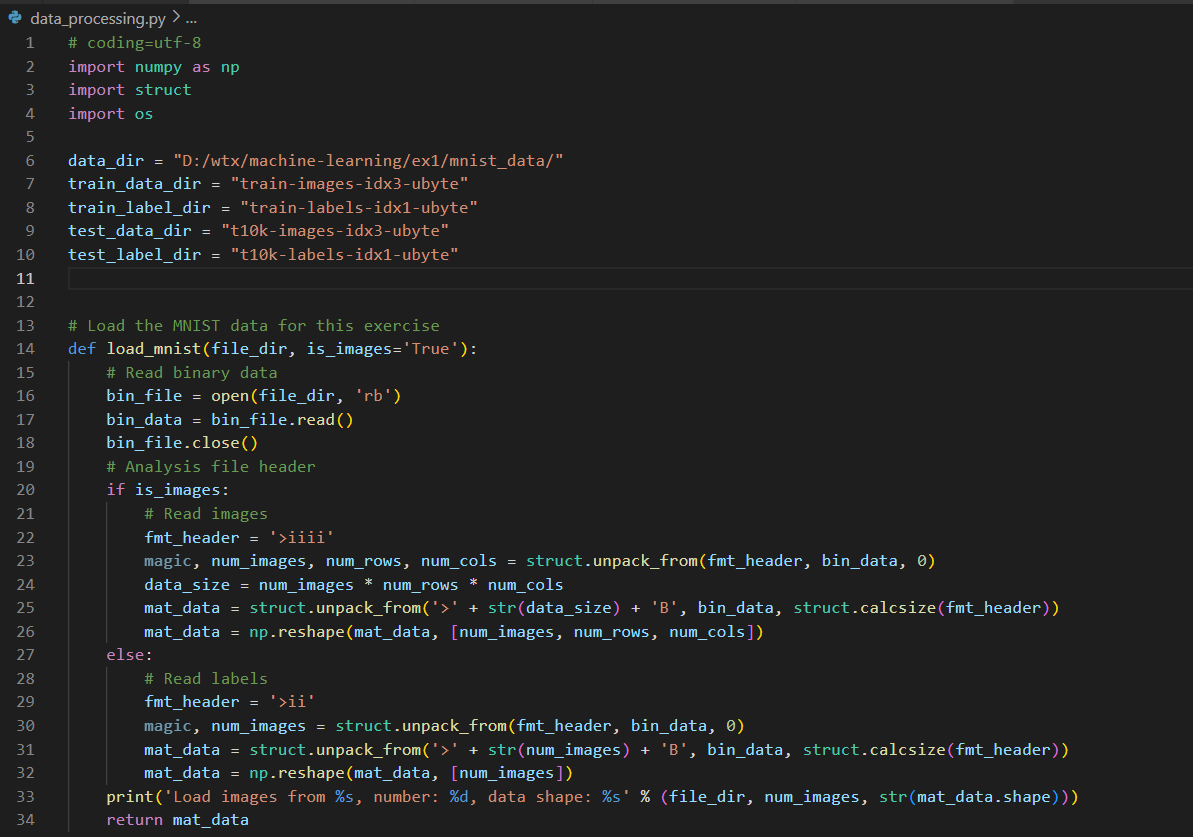
mian 函数中，首先是对数据的读取和正则化，由于 mnist 数据集中的图片为 28 \* 28，需要padding 到 32 \* 32。

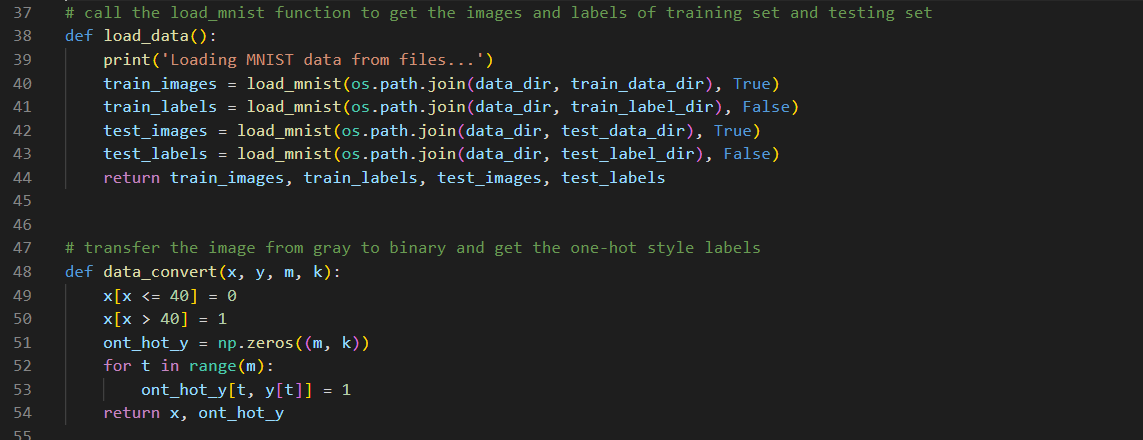
然后，将处理之后的数据输入 train 函数，开始训练。

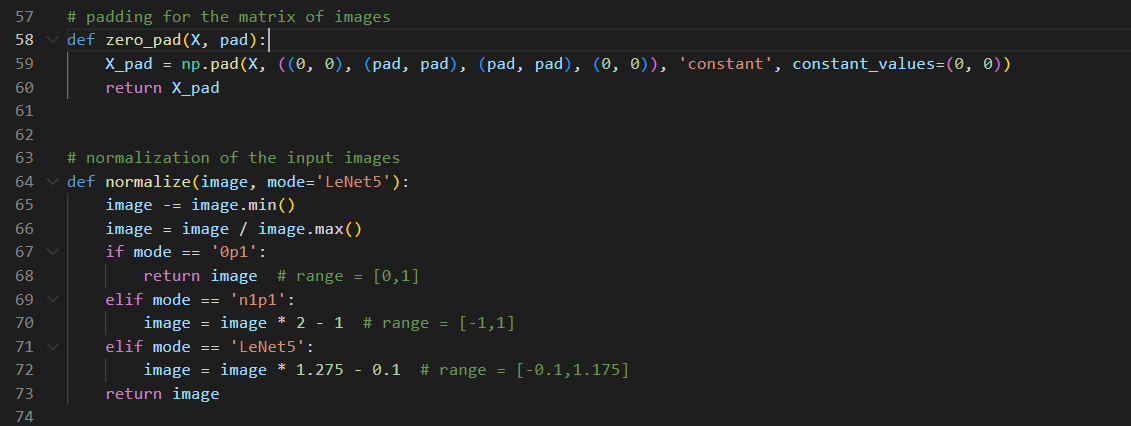
训练完成后，在测试集上测试训练网络的性能。



## 数据处理（data\_processing）

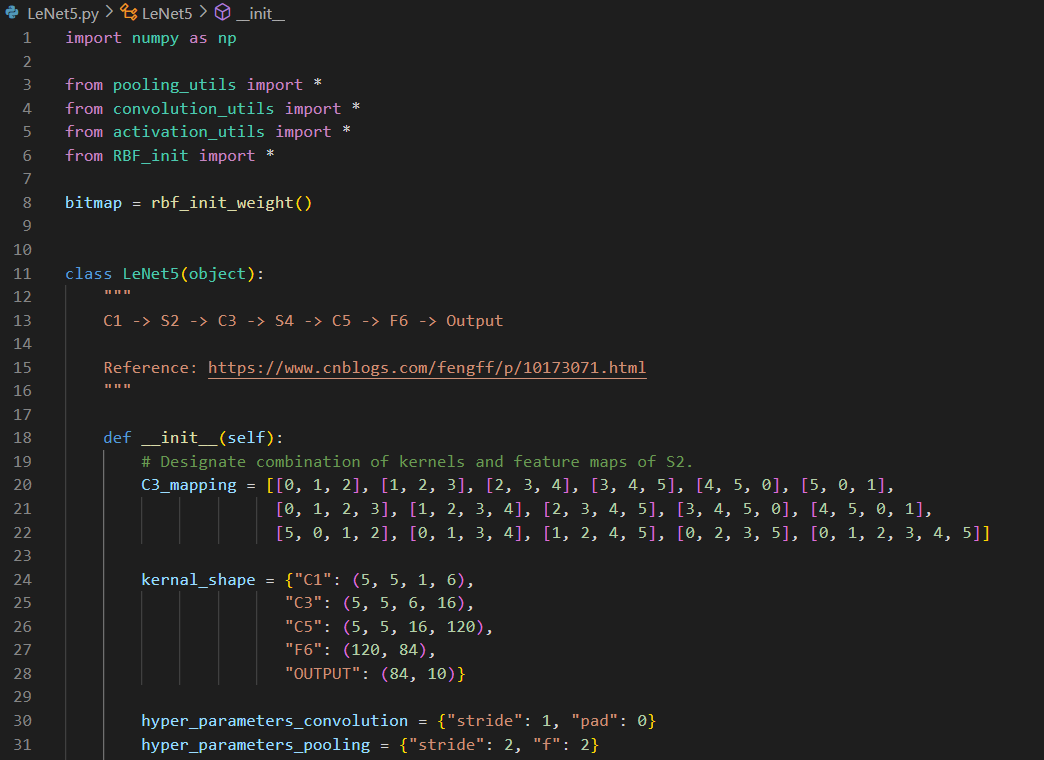


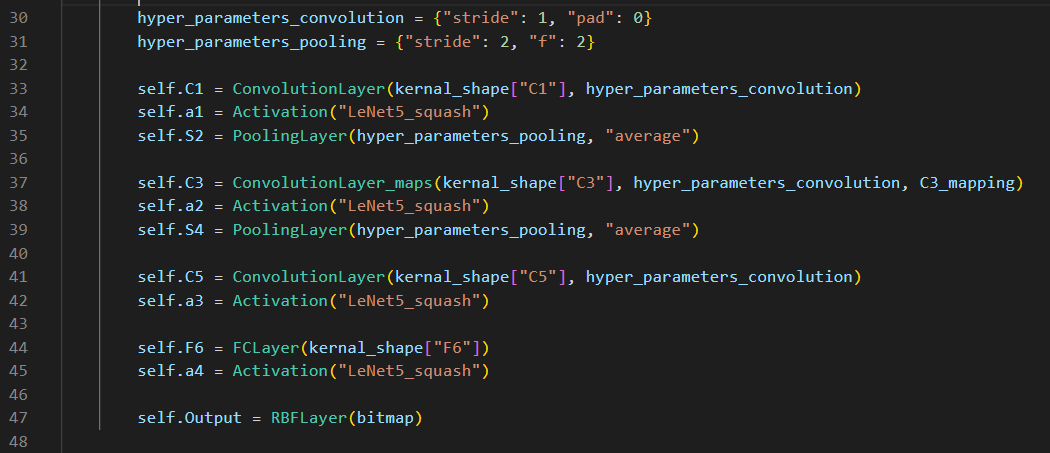


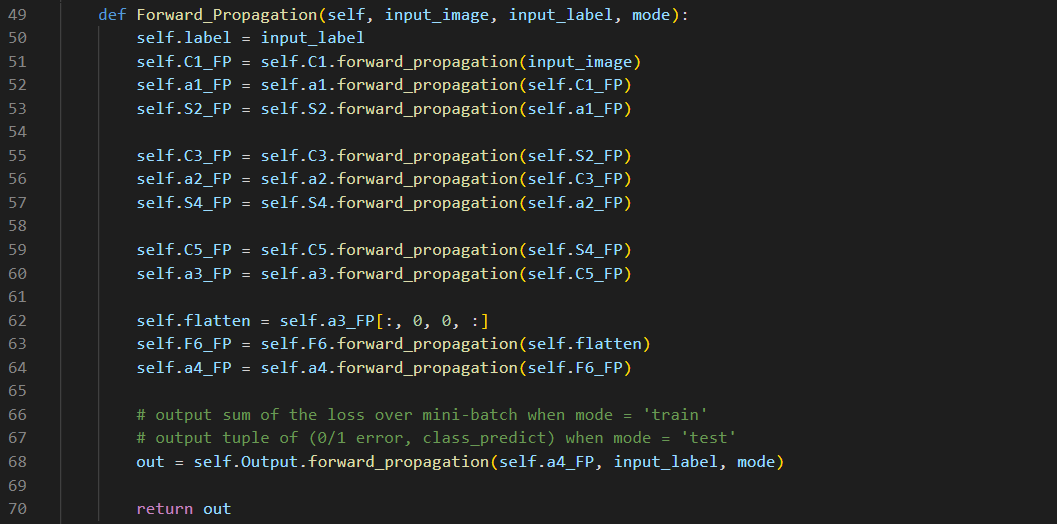


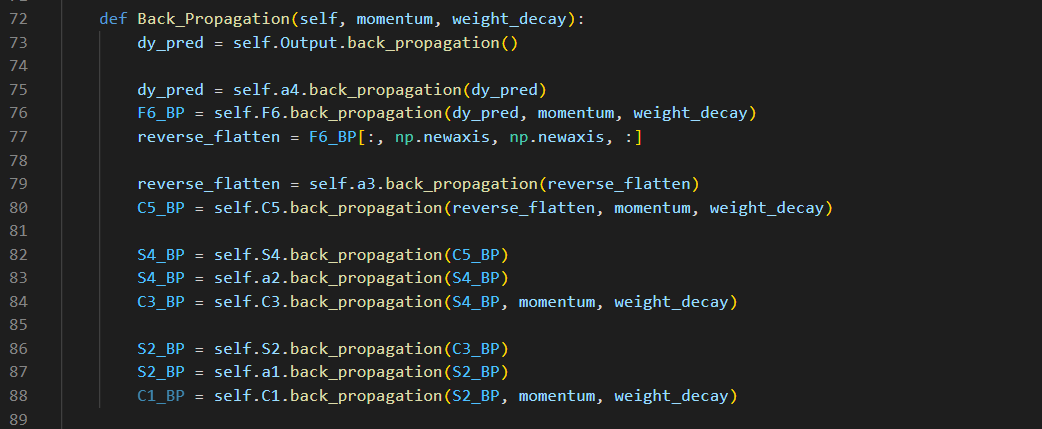
## LeNet5

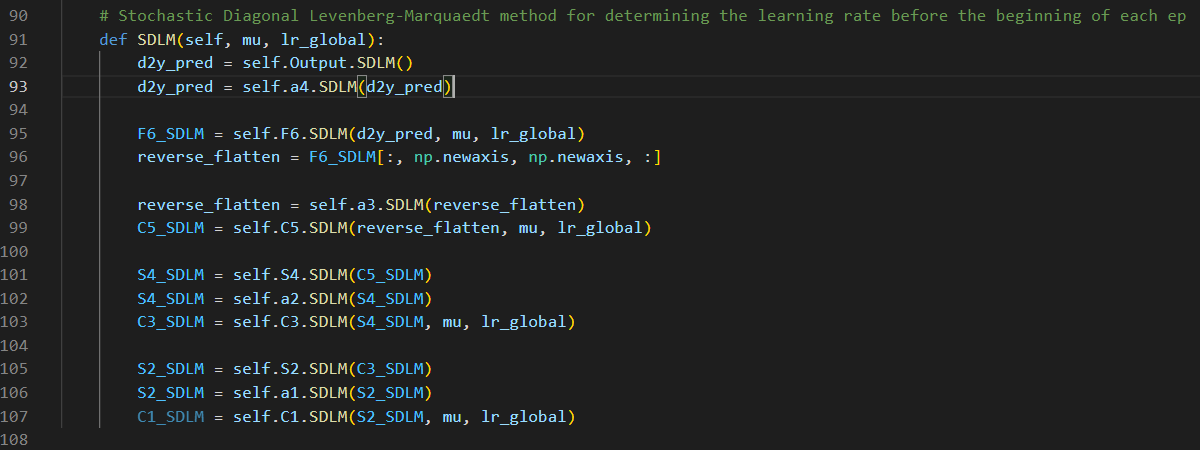
LeNet5 类的 Forward\_Propagation 和 Back\_Propagation 函数会调用每一层对应的 forward\_propagation 和 back\_propagation 函数，可以看到，层与层之间的参数会一层层的进行传递。通过 SDLM 算法进行计算得到的每一轮的学习率。



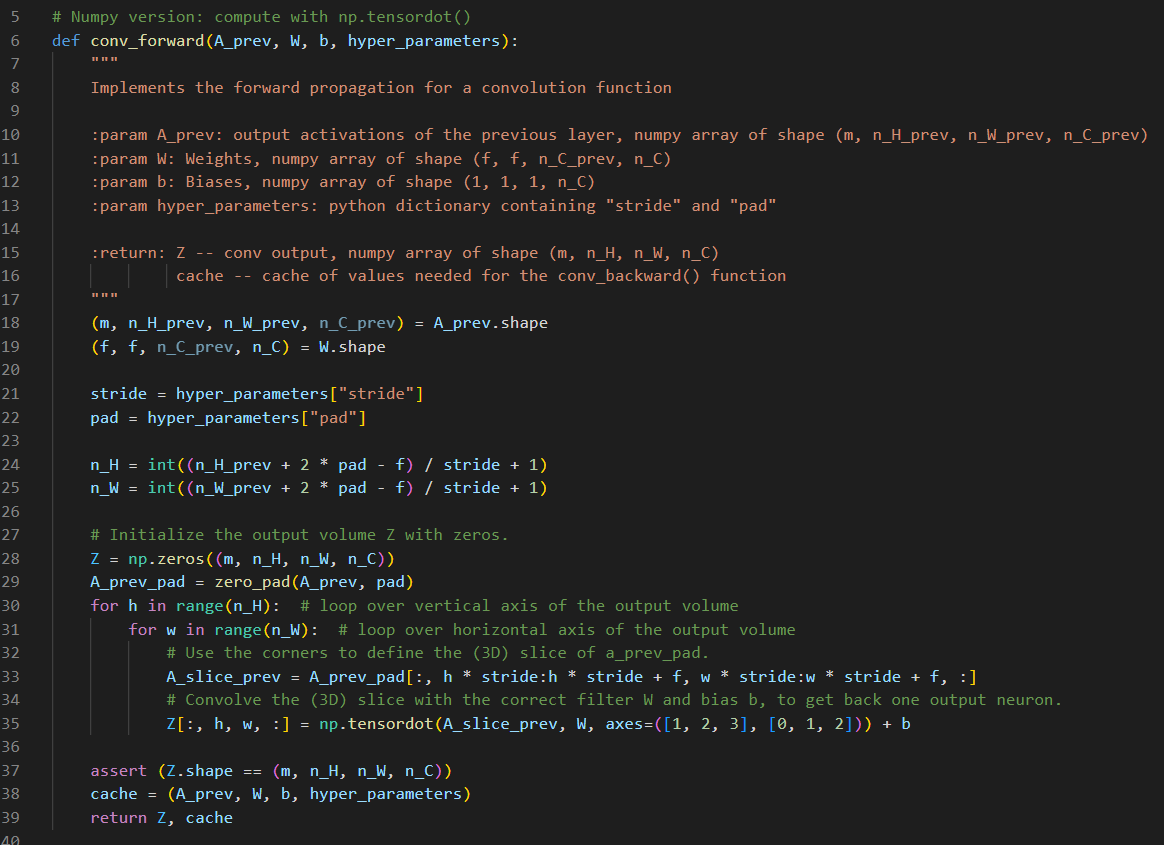


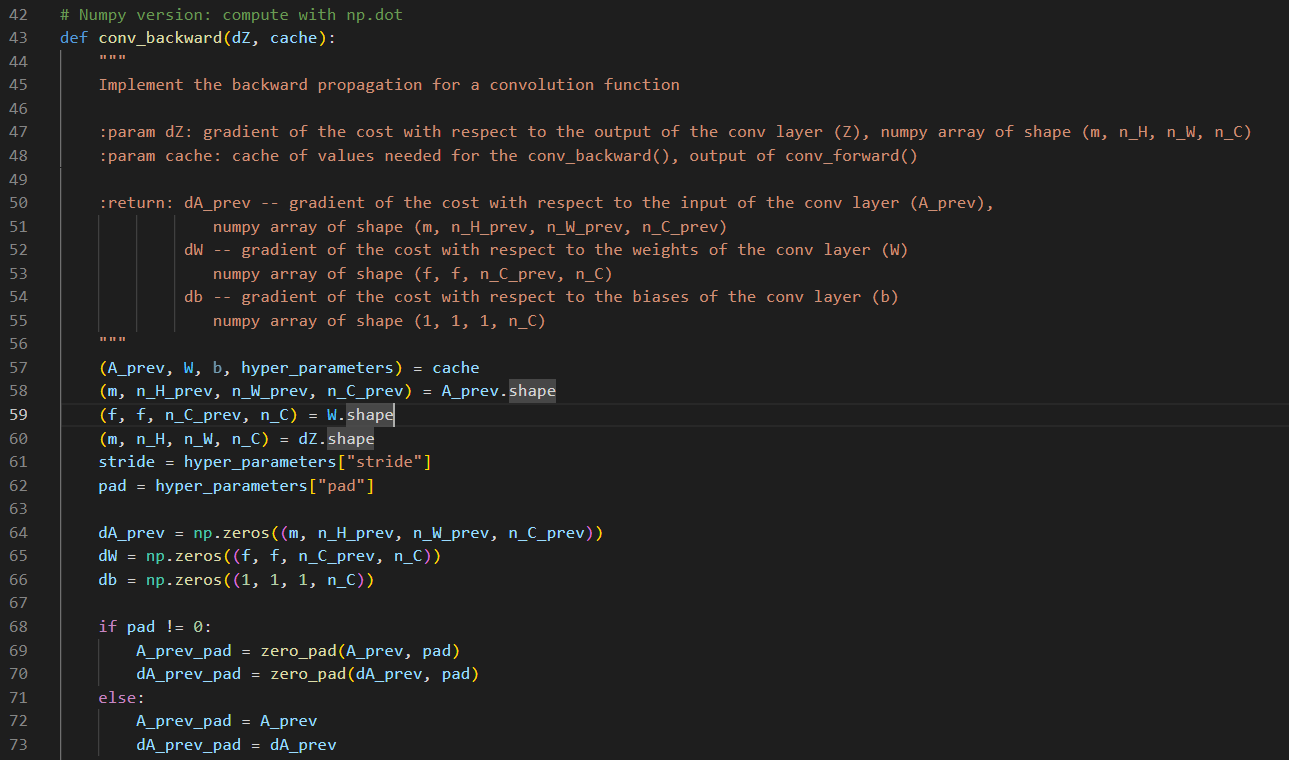


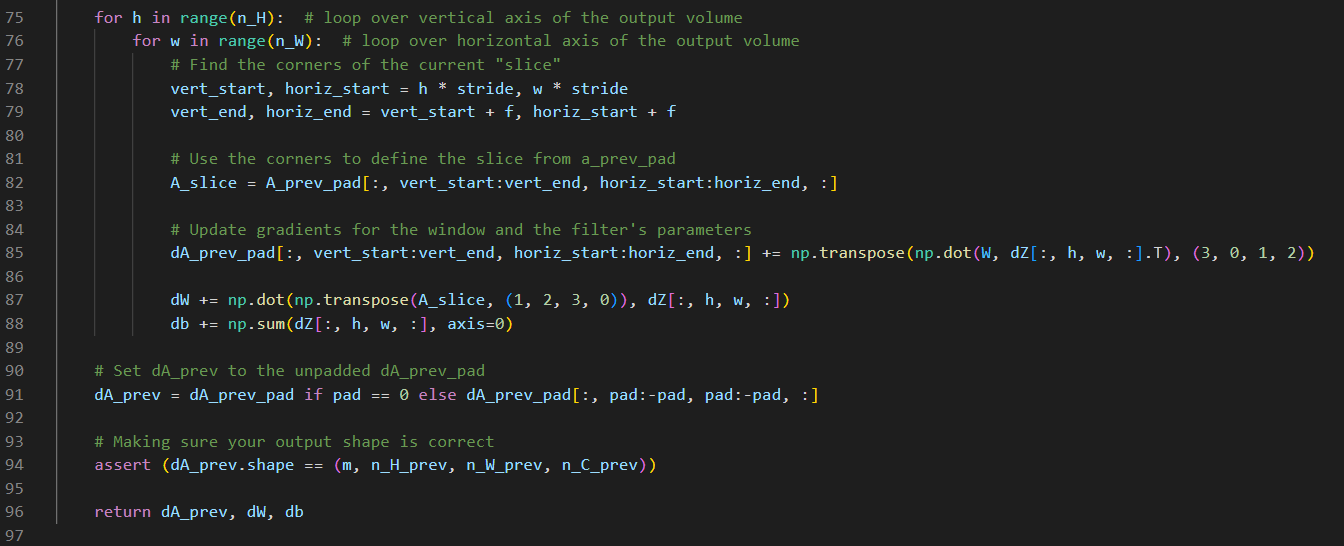


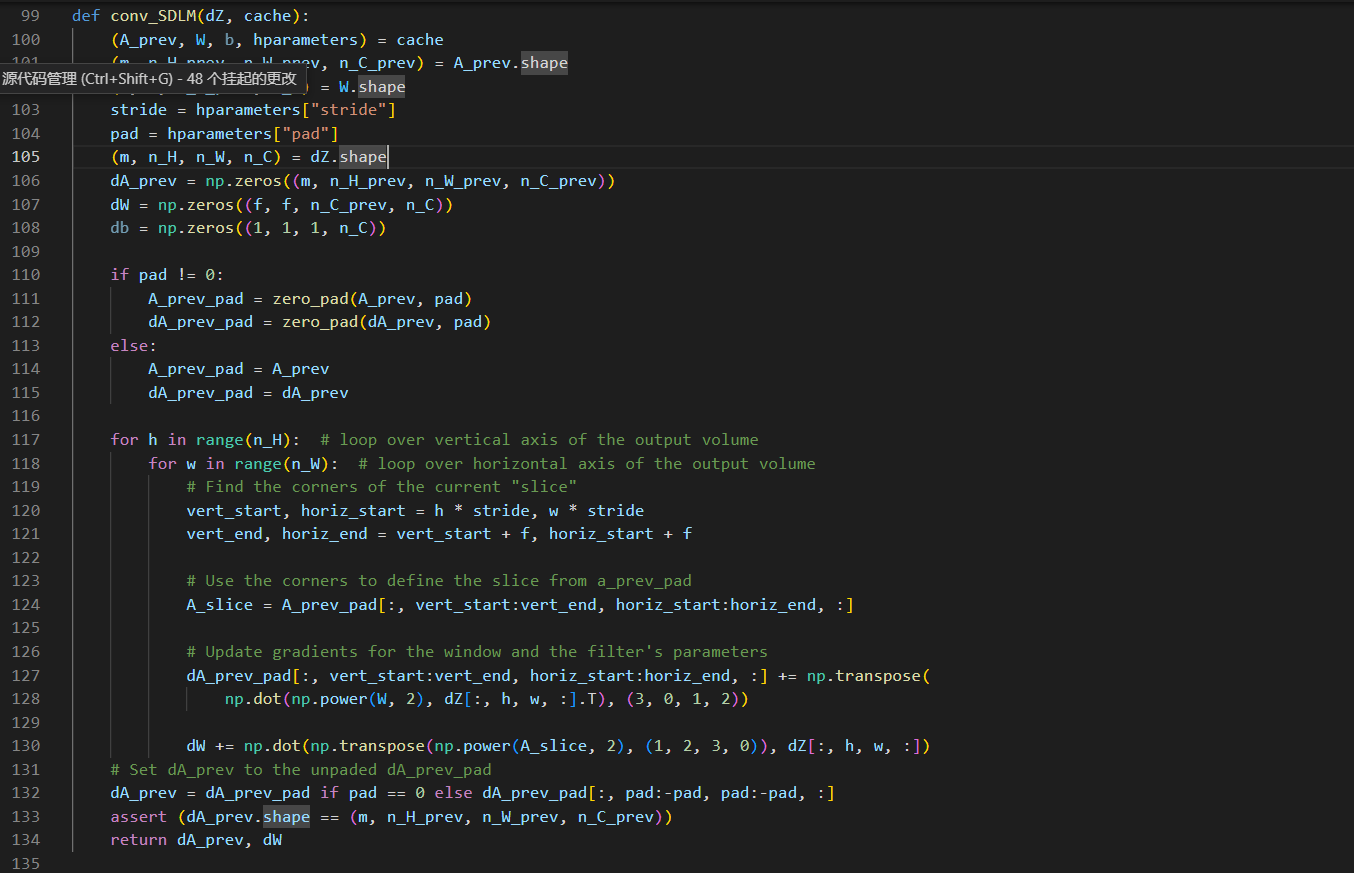


## 卷积层前向传播和后向传播（convolution\_utils）

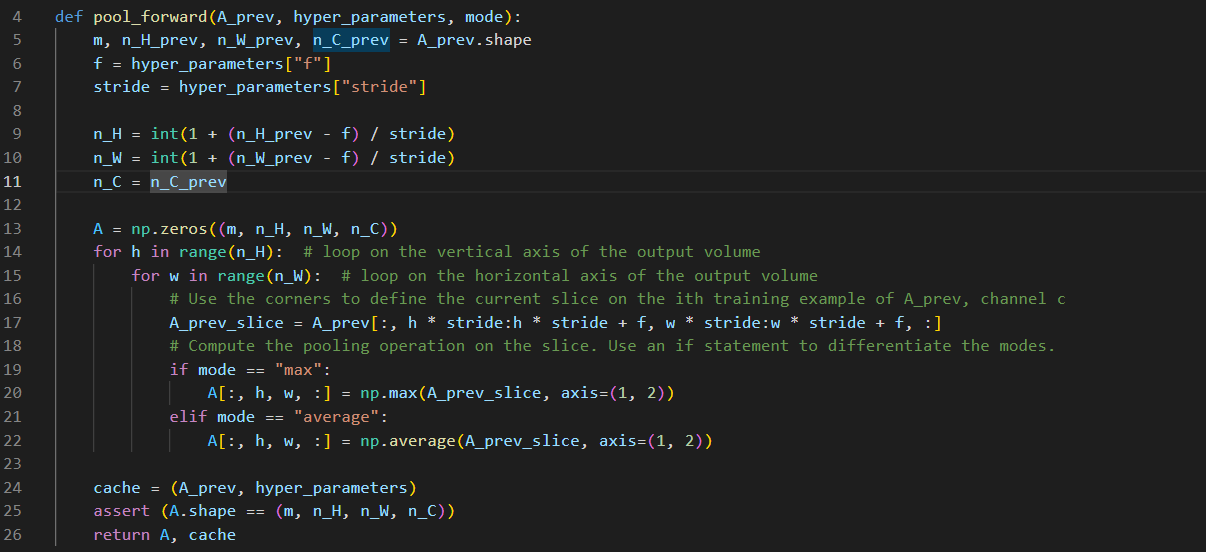


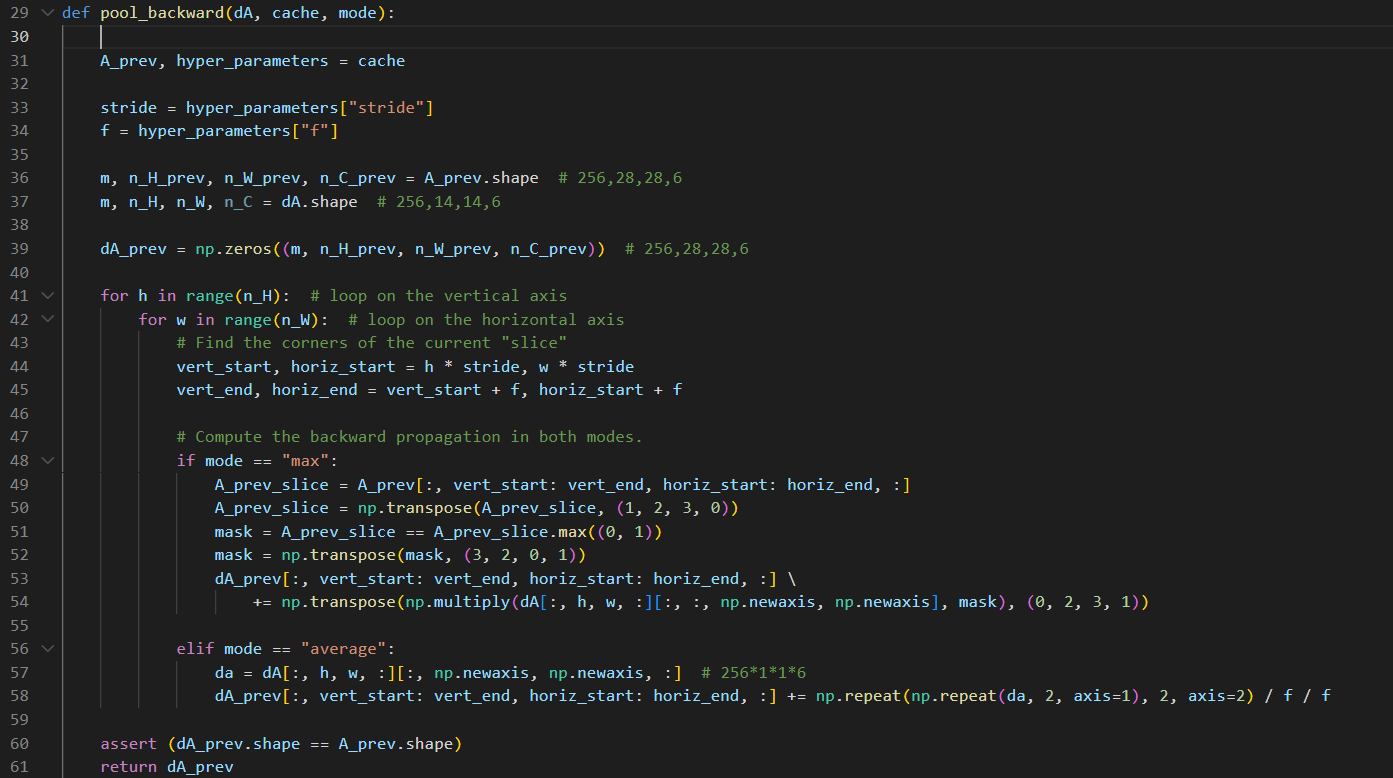


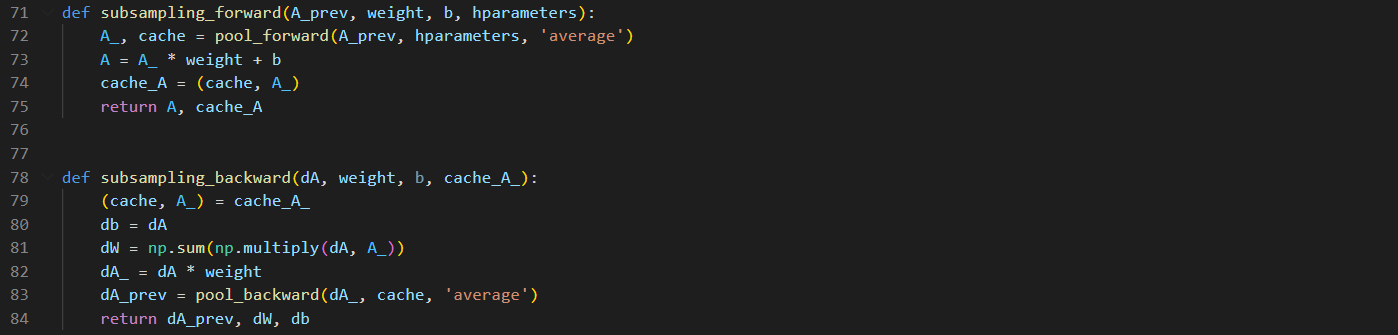




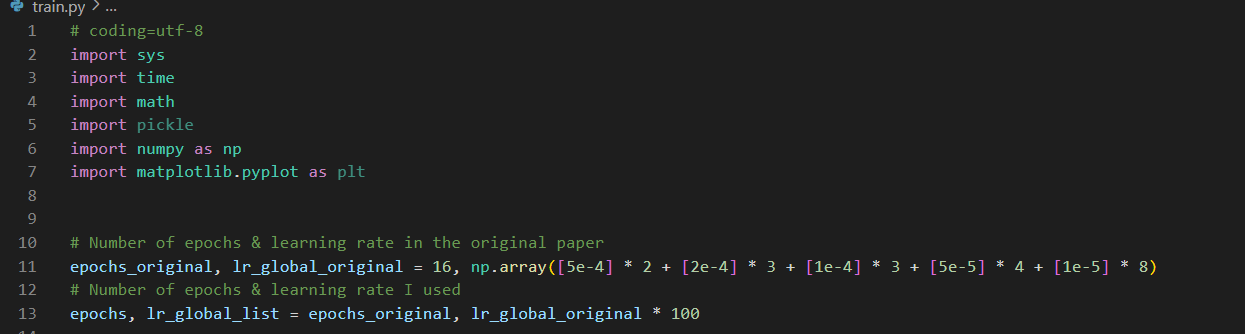
## 池化层前向传播和后向传播（pooling\_utils）

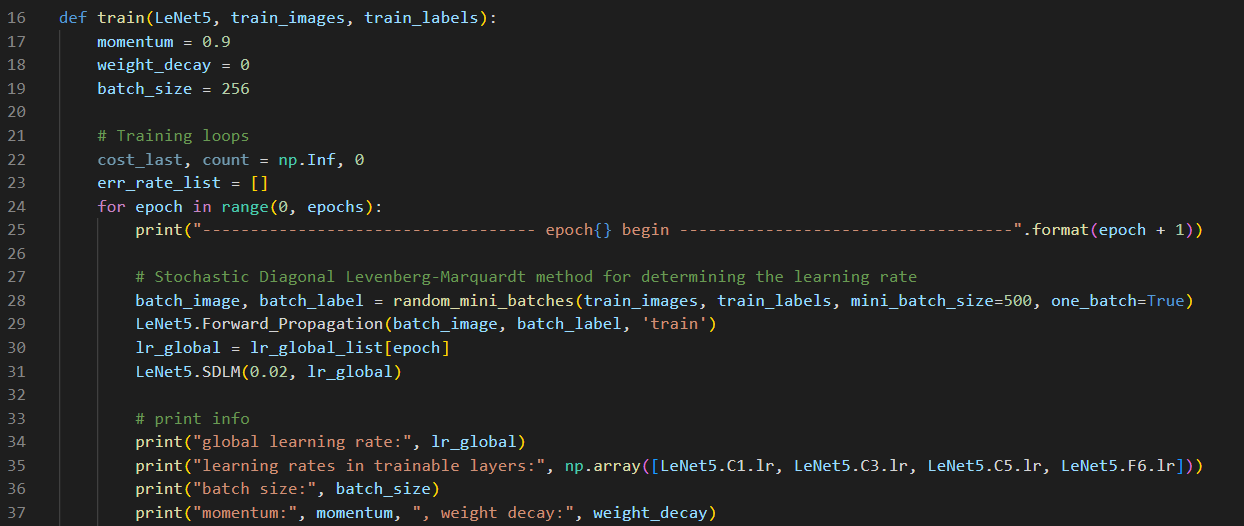


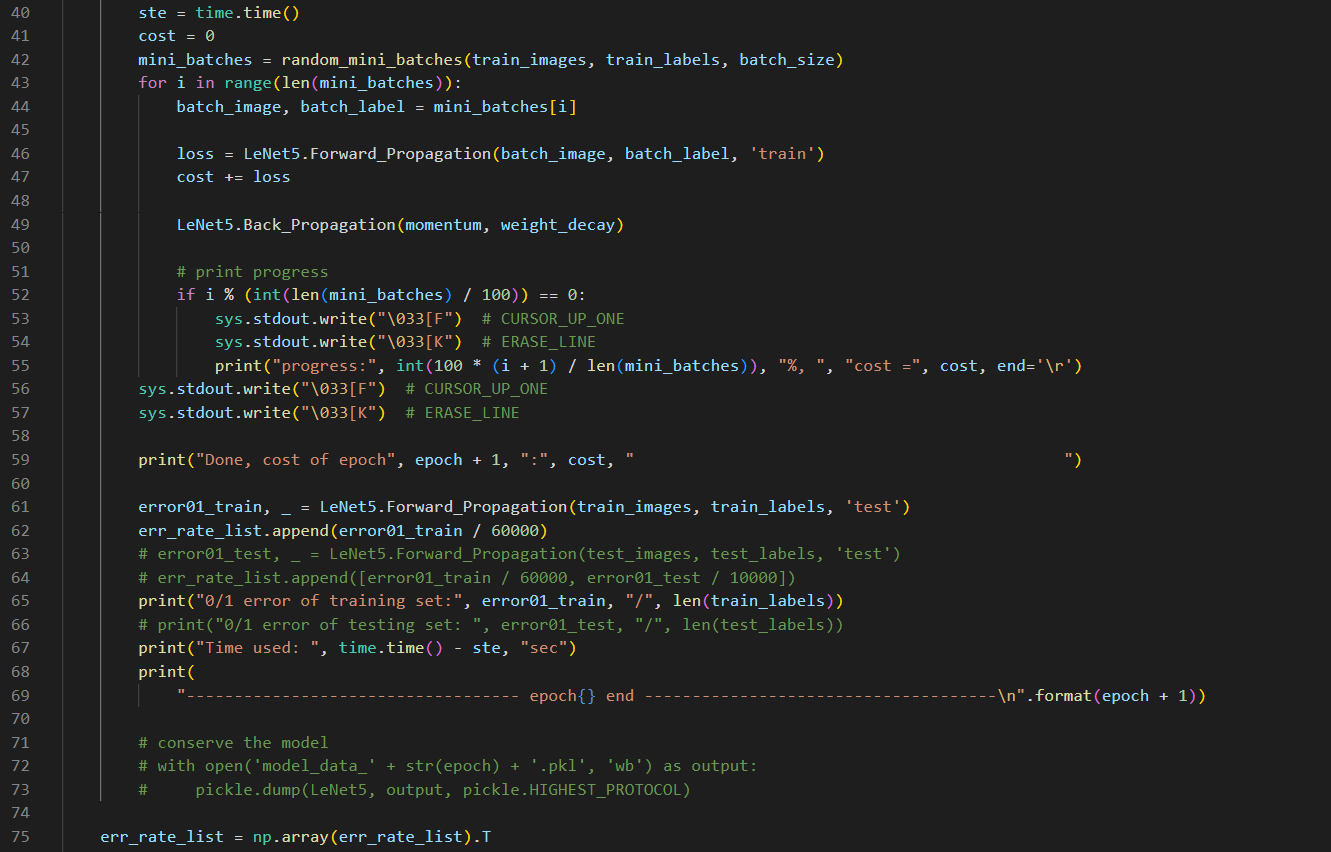


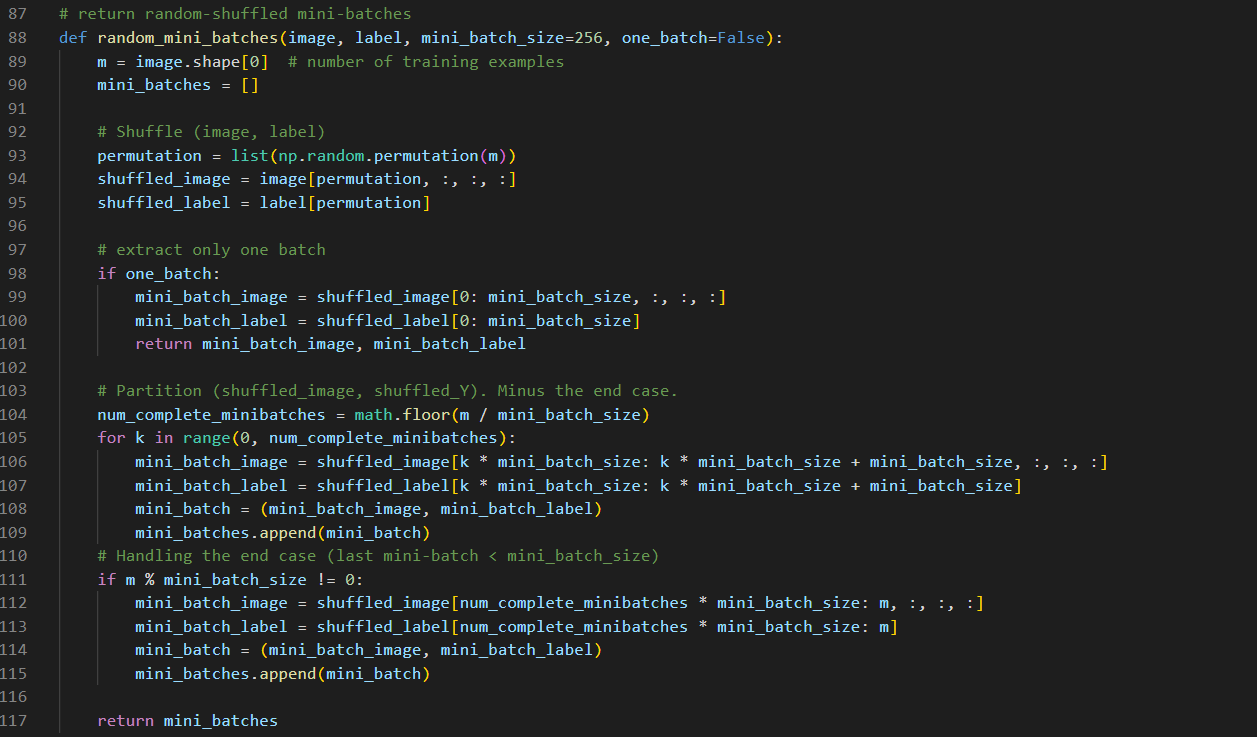


## 训练（train）



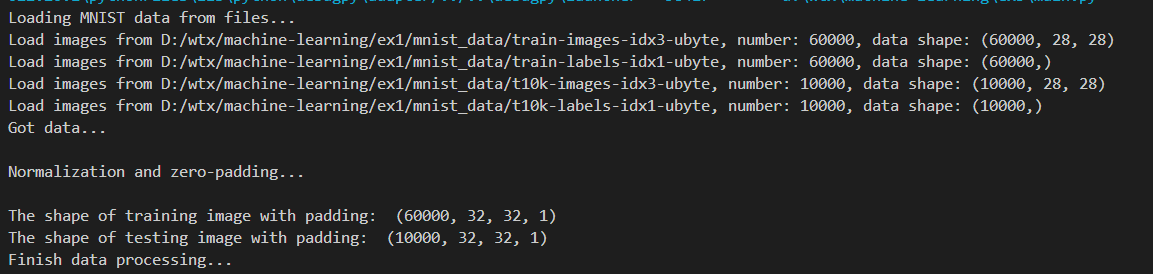




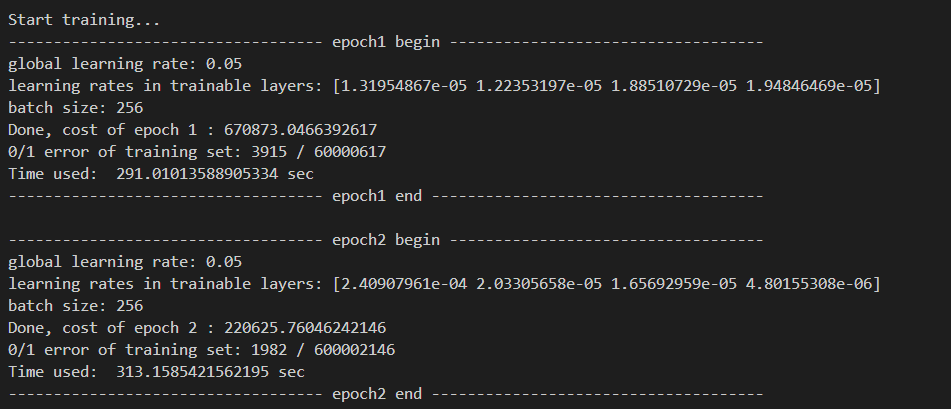


# 实验结果

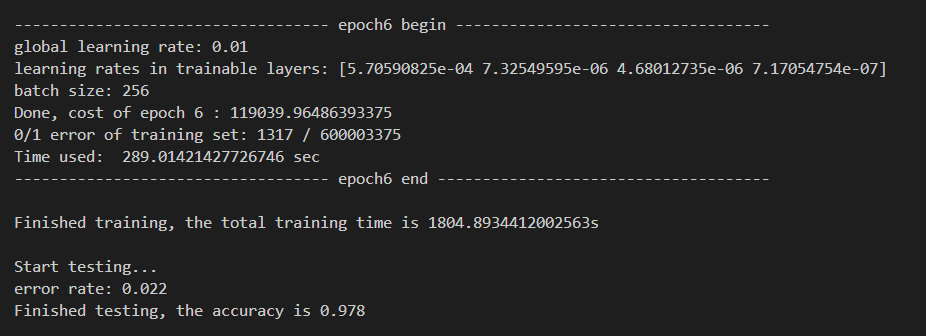
## 加载数据集并进行处理



## 训练模型



## 测试模型



正确率为97.8%，正确率较低的原因可能会训练模型时训练论述较低。但由于电脑配置较低，在轮数较高时会因为系统内存不足导致整个程序卡死，因此不得不降低轮数。（在只运行vcode的情况下，每一轮中，系统内存使用最高达到了96%）