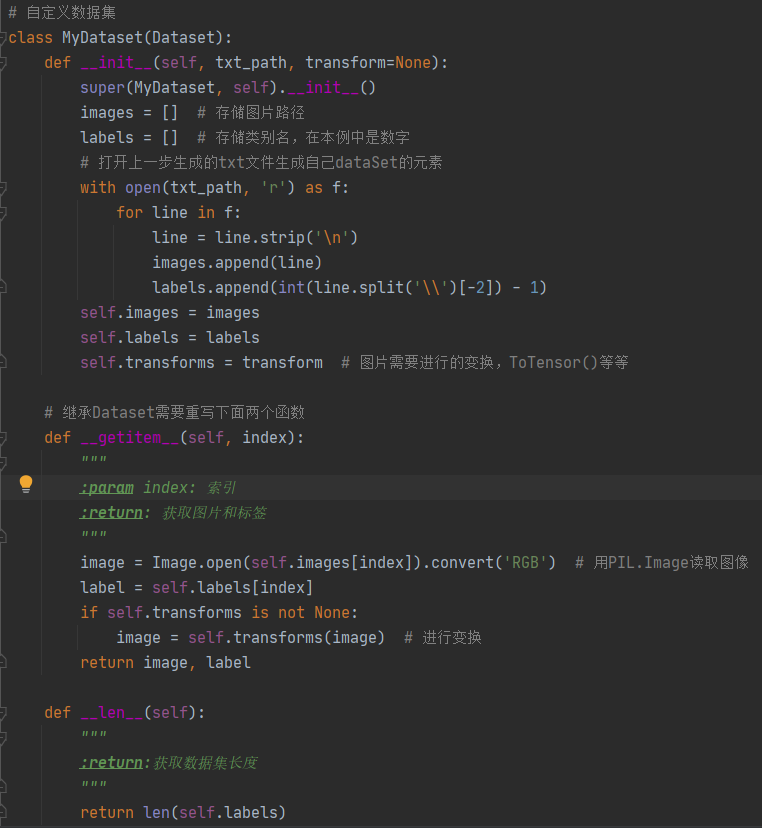
**智能系统原理与开发Lab1\_Part2**

18302010026 吕昌泽

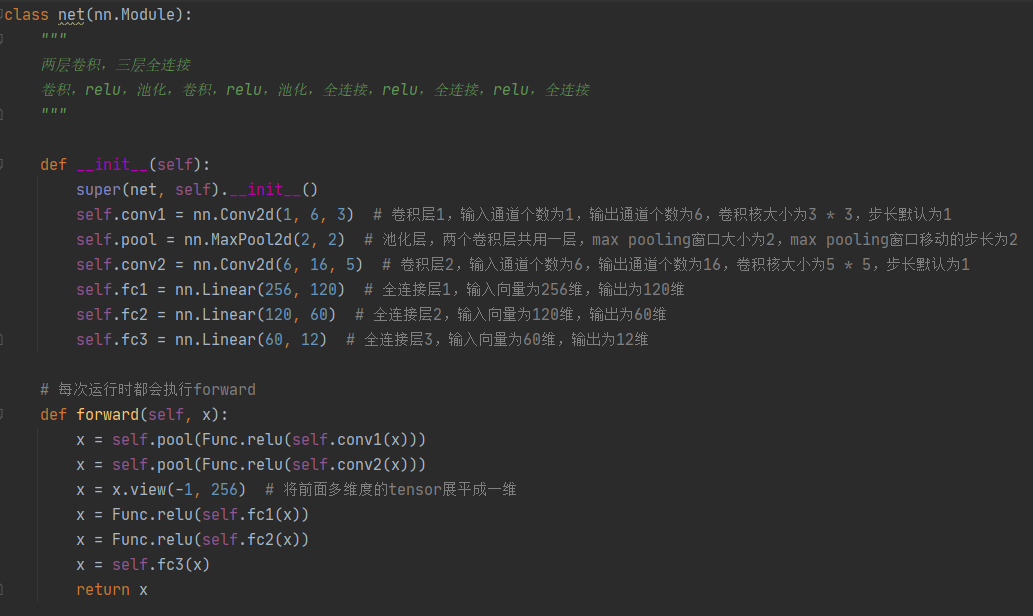
**代码基本架构**

自定义的数据集类：用于把创建自定义的训练集、测试集

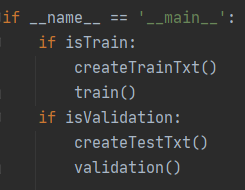


网络类：参数自己定。总共两层卷积，三层全连接。网络结构：卷积，激活，池化，卷积，激活，池化，全连接，激活，全连接，激活，全连接

图片的大小为28\*28，经过6个3\*3的卷积核，得到6张26\*26的feature map，通过2\*2且步长为2的池化，得到6张13\*13的feature map，再通过16个对每张feature map的5\*5的卷积核，得到16张9\*9的feature map，经过2\*2且步长为2的池化,得到16张4\*4的feature map，最后把16\*4\*4的向量展平成16\*16 = 256的一维向量。



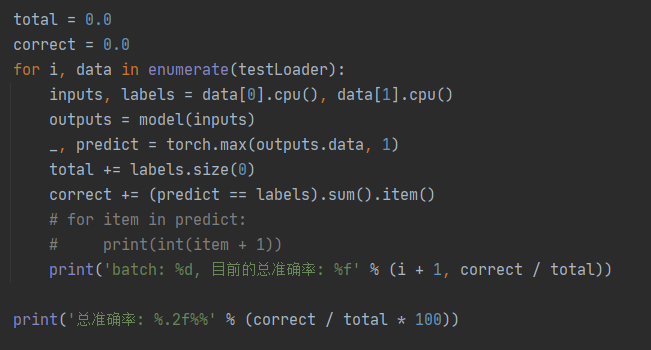
主方法：判断当前是在训练还是验证



Train方法：每个batch调整一次权重和bias，调整的过程用了框架，简化了很多。在训练后，用json对象保存得到的模型。



Validation方法根据已经保存的模型，进行验证并计算正确率

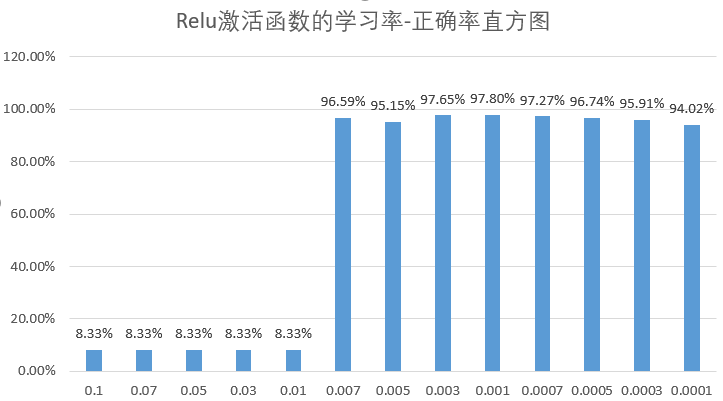


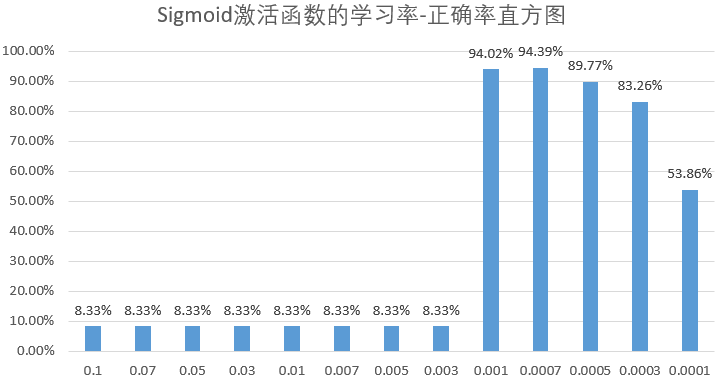
存取图片路径（训练或测试）的方法

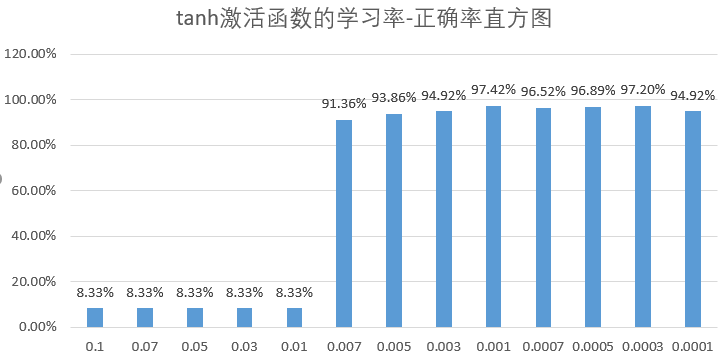


**超参的调整**

一、激活函数的选择：



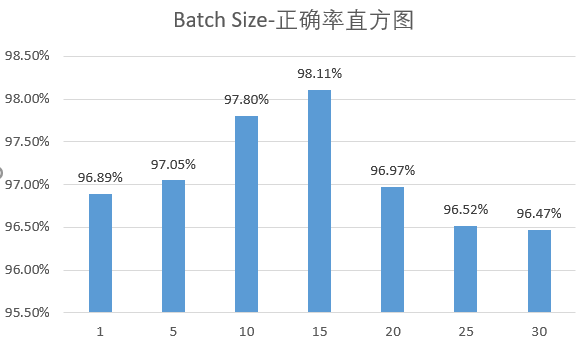




最高的正确率出现激活函数为relu，学习率为0.001的时候，达97.80%，为什么呢？

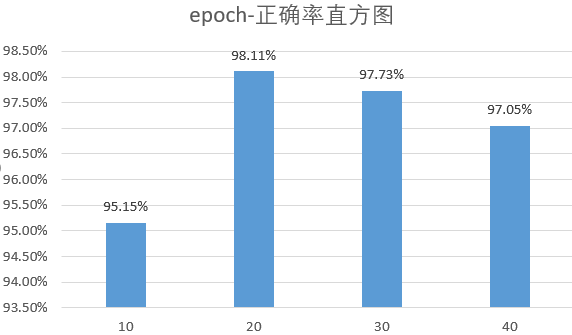
因为sigmoid和tanh是“饱和激活函数”，即左右两端导数的极限都趋近于0，而relu则是“非饱和激活函数”。使用“非饱和激活函数”的优势在于，能解决所谓的“梯度消失”，能加快收敛速度。但是relu的缺点在于，反向传播时一旦学习率没有设置好，使得某个神经元的ReLu 输入为负数，则会导致该神经元不再更新。

二、Batch Size的改进：



可以看出，当Batch Size的大小为15时，正确率最高。当batchSize太低时，梯度变来变去，难以使用一个全局的学习率进行调整，导致更新的权重，不准确，网络难以收敛；当batchSize太高时，与学习率不匹配。一般来说，batchSize设置较大时，学习率也要增大。

三、Epoch的改进



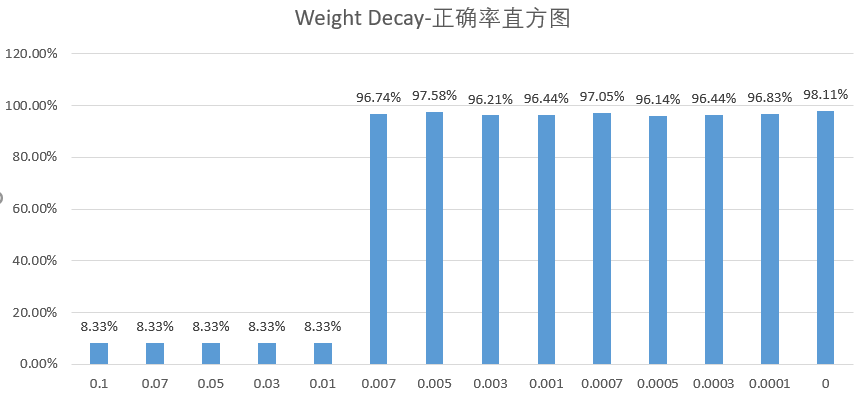
Epoch为20较好。太小无法收敛，太大过拟合。

四、优化方法的选择：



找不到使得SGD方法可以有效收敛的参数。Adam是自优化方法，利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。

五、对weight decay的改进：



Adam中的Weight decay参数，使用默认的0最好。

**网络结构改进**

一、Batch Normalize

Batch Normalize用于解决“Internal Covariate Shift”问题。在训练过程中，隐层的输入分布老是变来变去，Internal指的是深层网络的隐层，是发生在网络内部的事情，而covariate shift问题只发生在输入层。

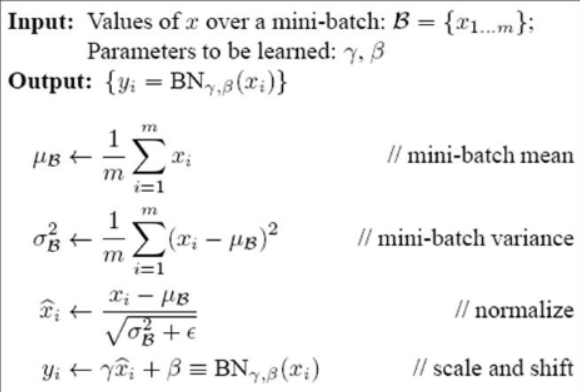
Batch Normalize的基本思想：让每个隐层节点的激活输入分布固定下来。BN 其实就是在做 feature scaling（特征缩放），即数据规范化，使得特征的范围具有可比性，是数据处理的预处理处理。它的目的是为了在训练的时候避免这种 Internal Covariate Shift 的问题，只是刚好也解决了 sigmoid 函数梯度消失的问题。

BN的优势在于：

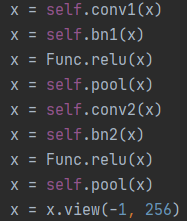
1、BN使得网络中每层输入数据的保持同分布，加速模型学习速度

2、BN使得模型对初始化方法和网络中的参数不那么敏感，简化调参过程，使得网络学习更加稳定

3、BN具有一定的正则化效果



采用nn.BatchNorm2d()方法。输入的参数为卷积层之后的通道个数。

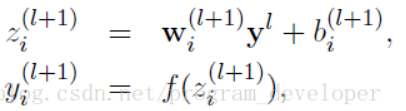


使用了Batch Normalize之后，正确率还是97%-98%，没有大变化。

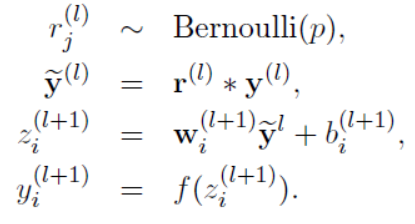
二、Dropout

在前向传播forward的过程中，增加一个dropout层。Droupout指在训练中，以一定的概率随机地“临时丢弃”一部分神经元，这是由于样本数据过少，防止过拟合而采用的trick。具体来讲，Dropout作用于每份小批量训练数据，由于其随机丢弃部分神经元的机制，相当于每次迭代都在训练不同结构的神经网络。

没有使用dropout时：

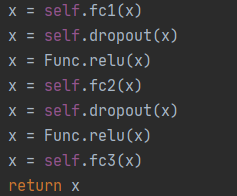


使用dropout以后：



Dropout的具体实现中，要求某个神经元节点激活值以一定的概率p被“丢弃”，即该神经元暂时停止工作。对于任意神经元，每次训练中都与一组随机挑选的不同的神经元集合共同进行优化，这个过程会减弱全体神经元之间的联合适应性，减少过拟合的风险，增加泛化能力。p一般设置为0.5或者0.3

Dropout层可以加在卷积层和全连接层的后面，但多见于放在fc 层后面，用来防止过拟合。



使用了Dropout之后，正确率还是97%-98%，没有大变化。

**对网络设计的理解**

CNN其实就是在BP的基础上，把全连接改为部分连接，然后再利用权值共享的技巧大量减少网络需要修改的权值数量。

filter对应的神经网络的概念其实是权重，filter矩阵和图像的单通道像素值进行点乘，最后将多个通道的结果求sum，作为一个filter的卷积结果。

Pooling层是一个特征选择、信息过滤的过程，也就是说我们损失了一部分信息。池化的常见方法分为max pooling，average pooling。采用max pooling，最主要的作用是提高空间不变性。因为一个像素不管最大值上在池化框内的哪个位置，都可以取到最大值。average pooling，就是将池化框内的值求平均，这样做的好处可以减少池化框大小带来的误差，更多的保留背景信息。而max pooling是减少卷积层权值参数的误差，更多的保留纹理信息。

CNN的反向传播在面试时解释。

**调整训练集以使正则化起到作用**

之前的训练集和测试集之比为510:110，由于测试集已经很多了，预测的性能比较准确，使得bn和dropout的小trick没办法体现成正确率的提升。

吧训练集和测试集调整为310:310，对不用trick和用了trick进行比较：

不用trick：97.15%

用了trick：97.82%

有略微的提升。