**基于神经网络的MNIST手写数字识别**

1. **实验目的**

* 掌握运用神经网络模型解决有监督学习问题
* 掌握机器学习中常用的模型训练测试方法
* 了解不同训练方法的选择对测试结果的影响

1. **实验内容**

**MNIST数据集**

本实验采用的数据集MNIST是一个手写数字图片数据集，共包含图像和对应的标签。数据集中所有图片都是28x28像素大小，且所有的图像都经过了适当的处理使得数字位于图片的中心位置。MNIST数据集使用二进制方式存储。图片数据中每个图片为一个长度为784（28x28x1，即长宽28像素的单通道灰度图）的一维向量，而标签数据中每个标签均为长度为10的一维向量。

**分层采样方法**

分层采样（或分层抽样，也叫类型抽样）方法，是将总体样本分成多个类别，再分别在每个类别中进行采样的方法。通过划分类别，采样出的样本的类型分布和总体样本相似，并且更具有代表性。在本实验中，MNIST数据集为手写数字集，有0~9共10种数字，进行分层采样时先将数据集按数字分为10类，再按同样的方式分别进行采样。

**神经网络模型评估方法**

通常，我们可以通过实验测试来对神经网络模型的误差进行评估。为此，需要使用一个测试集来测试模型对新样本的判别能力，然后以此测试集上的测试误差作为误差的近似值。两种常见的划分训练集和测试集的方法：

留出法（hold-out）直接将数据集按比例划分为两个互斥的集合。划分时为尽可能保持数据分布的一致性，可以采用分层采样（stratified sampling）的方式，使得训练集和测试集中的类别比例尽可能相似。需要注意的是，测试集在整个数据集上的分布如果不够均匀还可能引入额外的偏差，所以单次使用留出法得到的估计结果往往不够稳定可靠。在使用留出法时，一般要采用若干次随机划分、重复进行实验评估后取平均值作为留出法的评估结果。

k折交叉验证法（k-fold cross validation）先将数据集划分为k个大小相似的互斥子集，每个子集都尽可能保持数据分布的一致性，即也采用分层采样（stratified sampling）的方法。然后，每次用k-1个子集的并集作为训练集，余下的那个子集作为测试集，这样就可以获得k组训练集和测试集，从而可以进行k次训练和测试。最终返回的是这k个测试结果的均值。显然，k折交叉验证法的评估结果的稳定性和保真性在很大程度上取决于k的取值。k最常用的取值是10，此外常用的取值还有5、20等。

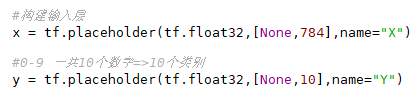
三、实验方法设计

介绍实验中程序的总体设计方案、关键步骤的编程方法及思路，主要包括:

1. 模型构建的程序设计（伪代码或源代码截图）及说明解释 （10分）

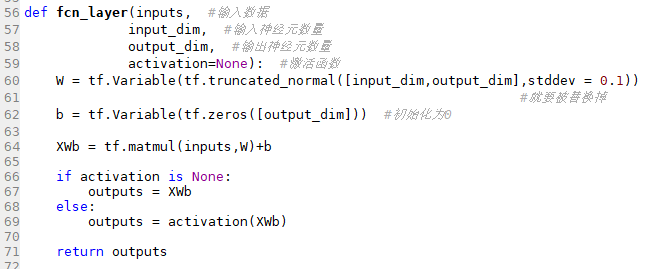
在这里训练集和测试集是经过打乱的，都是从mnist.train里划分出来的。用的验证集是mnist.validation中的。

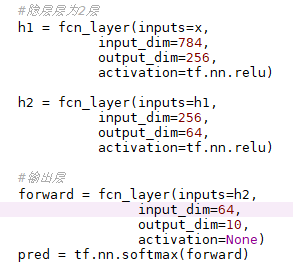


图片数据中每个图片为一个长度为784（28x28x1，即长宽28像素的单通道灰度图）的一维向量，而标签数据中每个标签均为长度为10的一维向量。故占位符（即输入层x）设置为：

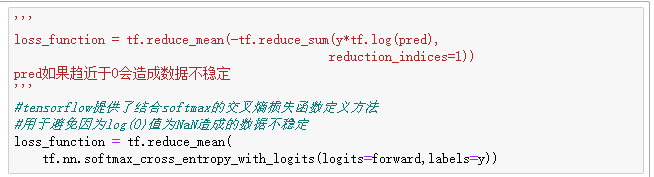
为了满足老师所说的要求，即模型准确率达到97%以上，选择隐藏层为2层，神经元分别为256和64。这里激活函数选择的是relu。

在这里之前，我定义了一个层函数，为了方便：

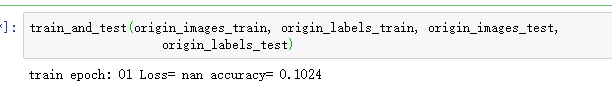




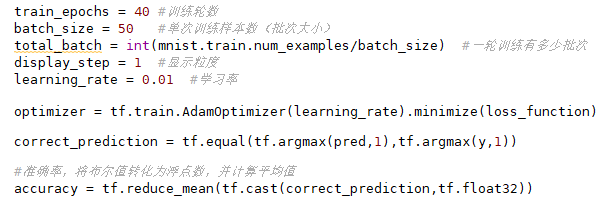
交叉熵损失函数是刻画的是两个概率分布之间的距离，交叉熵越小，两个概率的分布越接近。



以下是使用红色部分交叉熵会出现的结果：



训练时的超参数，选择的优化器和定义的准确率如下：



1. 模型迭代训练的程序设计（伪代码或源代码截图）及说明解释 （10分）

模型迭代40轮，每轮全部数据集，但是按每批次50个样本进行训练，

这里迭代训练模型用了老师给的batch\_iter（）函数，是因为在hold\_on和k\_fold中，我所分割的训练集和测试集有一定的规律，每一块都是按照标签分布的。

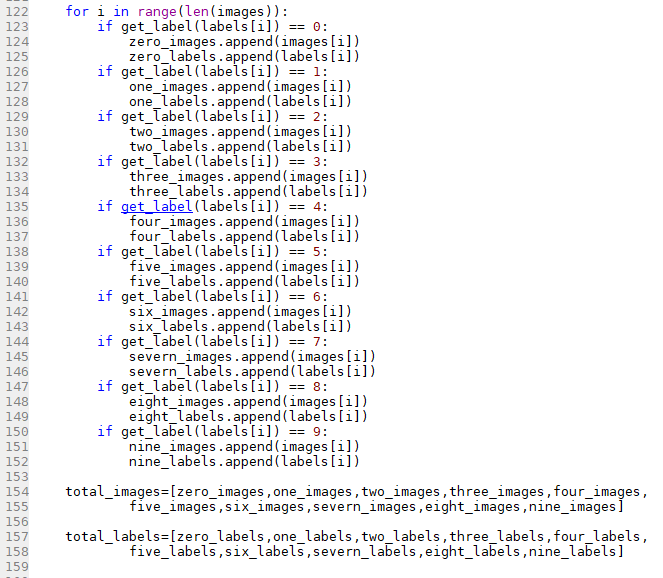


1. 模型训练过程中周期性测试的程序设计（伪代码或源代码截图）及说明解释（周期性测试指的是每训练n个step就对模型进行一次测试，得到准确率和loss值）（10分）

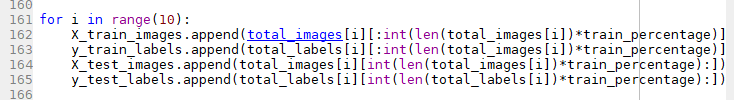
请见上文

1. 分层采样的程序设计（伪代码或源代码截图）及说明解释 （10分）

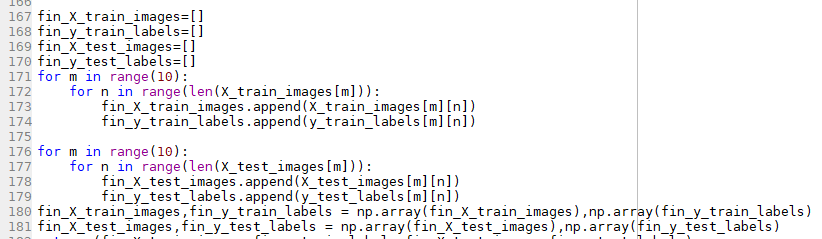
首先我是用get\_label（）函数获取图像的标签，按照标签把图像进行分类。部分代码：



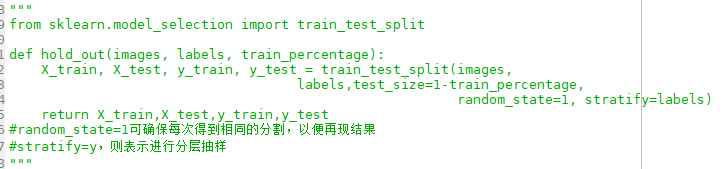
然后进行分离，前train\_percentage属于训练集，后半部分属于测试集。



但是我这里的训练集和测试集都处于多层嵌套的列表，需要将其转为一维列表，最后转为array，用于训练和测试：



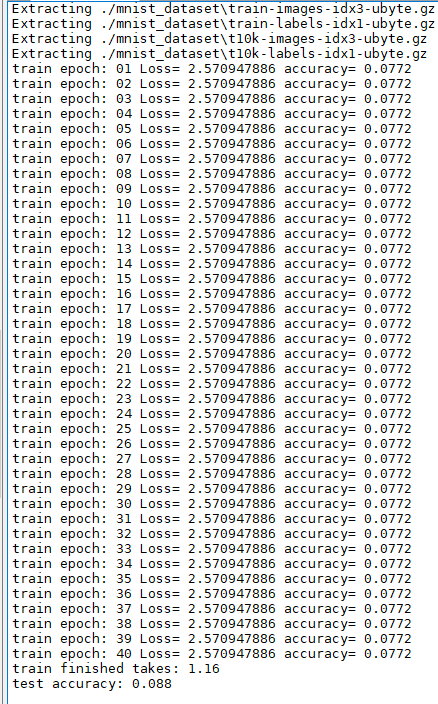
其实python有自带的库，见如下：



但我还是造轮子了

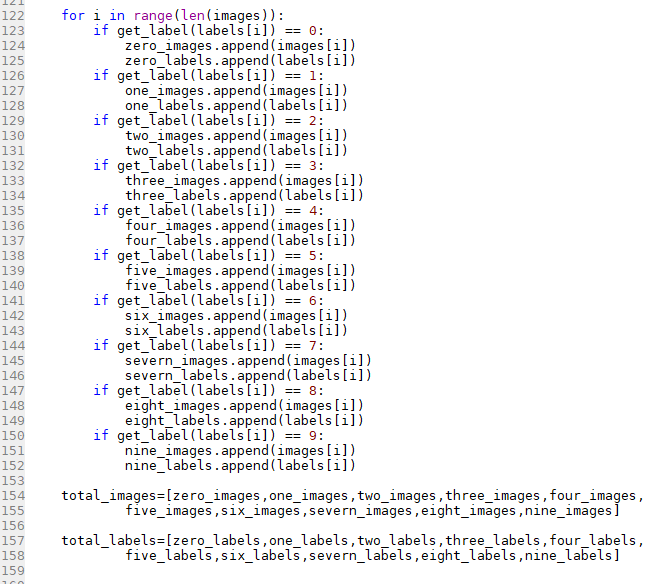
1. k折交叉验证法的程序设计（伪代码或源代码截图）及说明解释 （10分）

这个是最要命的，我有大部分时间卡死在了将训练数据集从多层嵌套列表转为一维列表上，因为逻辑有些转不过来了。以下是对于训练数据集没有脱离多层嵌套训练的模型结果：



可以知道，这里训练集有可能出了问题。

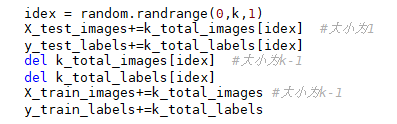
脱离多层嵌套列表的方法如下，首先对标签进行分类与上面一样：



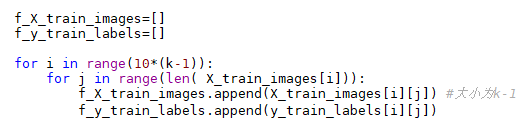
随后将每一个标签进行k分离，按照 i\*len(images)/k：(i+1)\*len(images)/k的规律进行分离：



随后利用random.randrange()函数随机选取每一个标签中的一份作为测试集，剩余部分作为训练集：



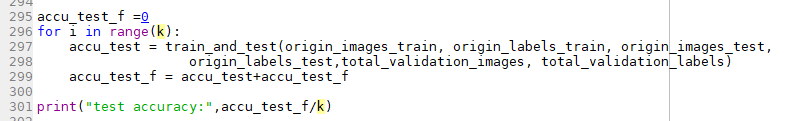
需要注意的是这里测试集已经是一维列表了，而训练集还是多层嵌套的列表，长度为10\*（k-1），里面还包含了各个标签的长度：所以它转为一维列表如下：



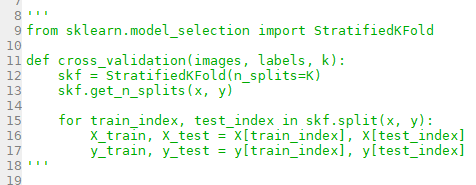
最后全部转为array



随后需要对模型训练k次，取平均值，这是让train\_and\_test()返回accu\_test,随后代码如下：



其实这个也有自带的python库，如下：



但我还是造轮子了

**四、实验结果展示**

展示程序界面设计、运行结果及相关分析等，主要包括：

1. 模型在测试集下的准确率（输出结果并截图）（10分）

三层隐藏层：第一层神经元为256，第二层神经元为64，第三层32

激活函数：relu

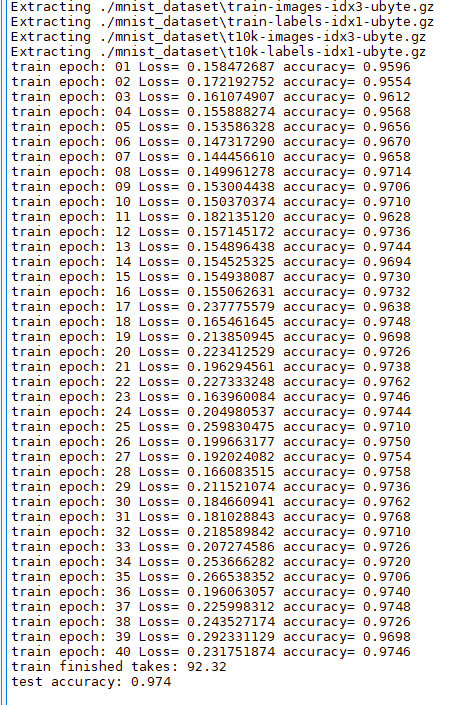
交叉熵损失函数：softmax\_cross\_entropy\_with\_logits

训练轮数：40

批次大小：50

学习率：0.01

优化器选择的是：Adam



1. 不同模型参数（隐藏层数、隐藏层节点数）对准确率的影响和分析 （10分）

激活函数：relu

交叉熵损失函数：softmax\_cross\_entropy\_with\_logits

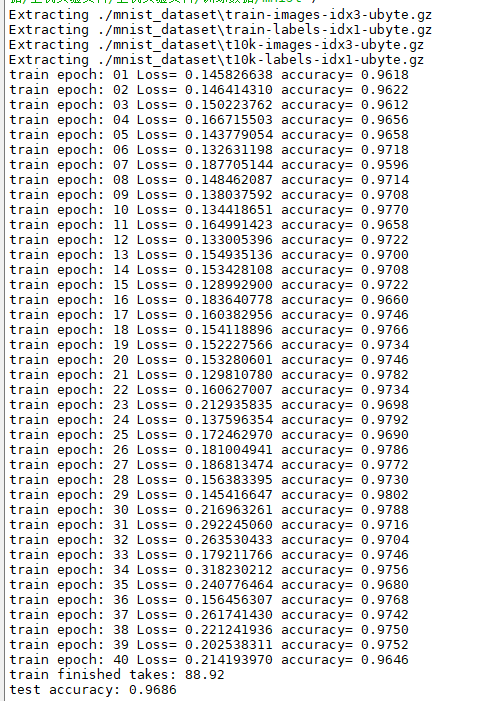
训练轮数：40

批次大小：50

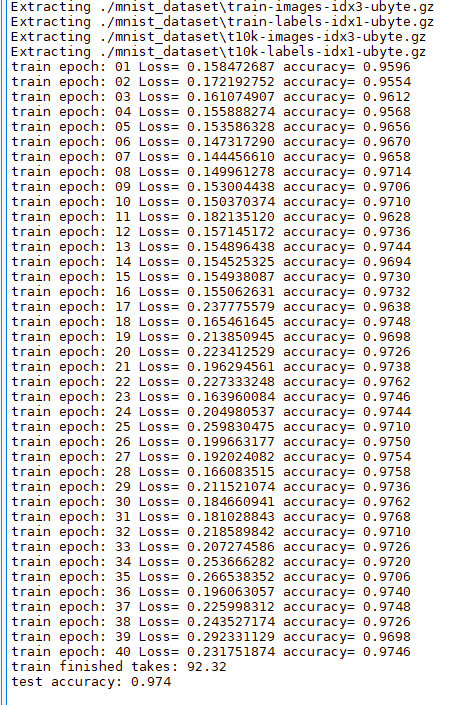
学习率：0.01

优化器选择的是：Adam

两层隐藏层：第一层神经元为256，第二层神经元为64



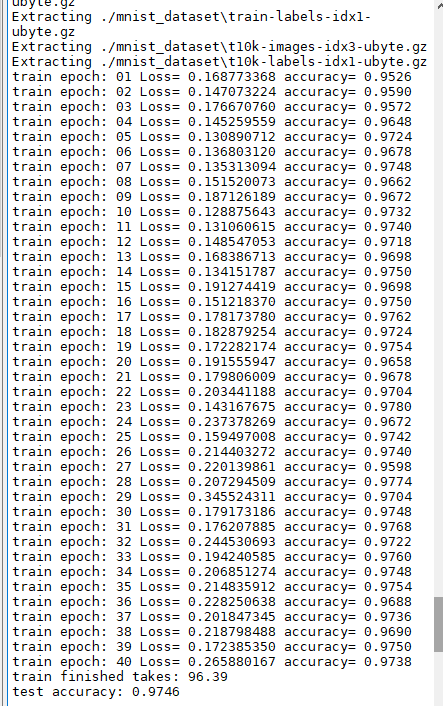
三层隐藏层：第一层神经元为256，第二层神经元为64，第三层神经元为32



隐藏层增加，准确率升高，所花费的时间变长

多个隐藏层其实是对输入特征多层次的抽象，最终的目的就是为了更好的线性划分不同类型的数据，但层数越多，参数也会爆炸式增多。到了一定的层数，再加深隐藏层，分类效果增强会越来越不明显。

在三层隐藏层的基础上，改掉第三层神经元数目为56：

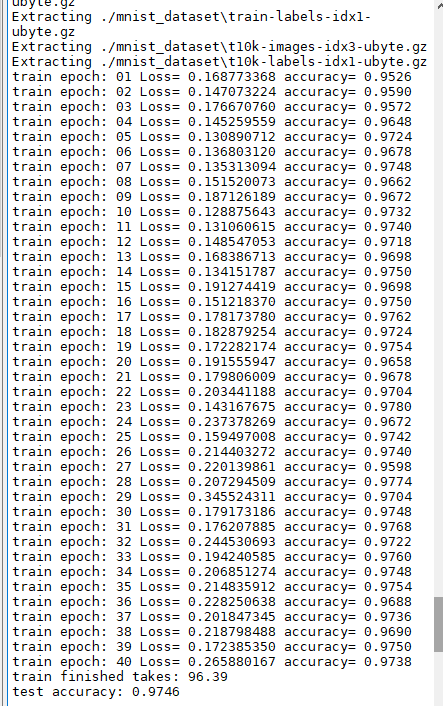


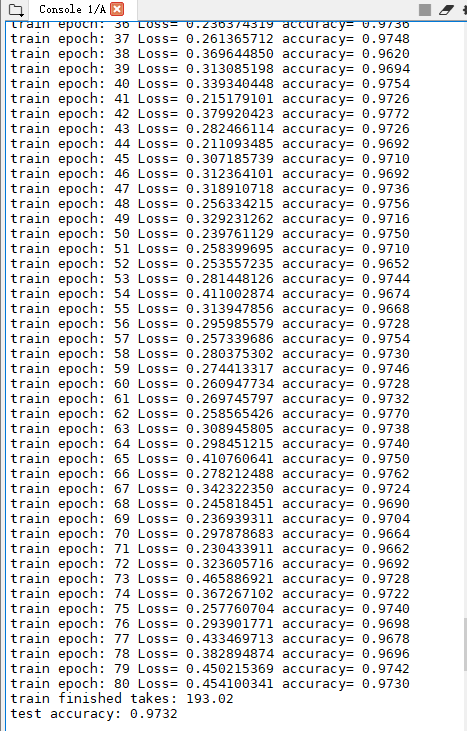
时间花费更长，准确率提了一丢丢

神经元越多，学习到的东西就越来越像样本，可能造成过拟合。

1. 不同训练参数（batch size、epoch num、学习率）对准确率的影响和分析 （10分）

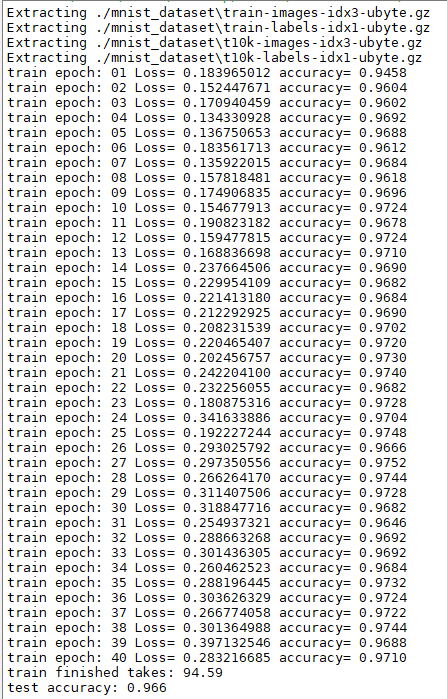
epoch num：由40->80

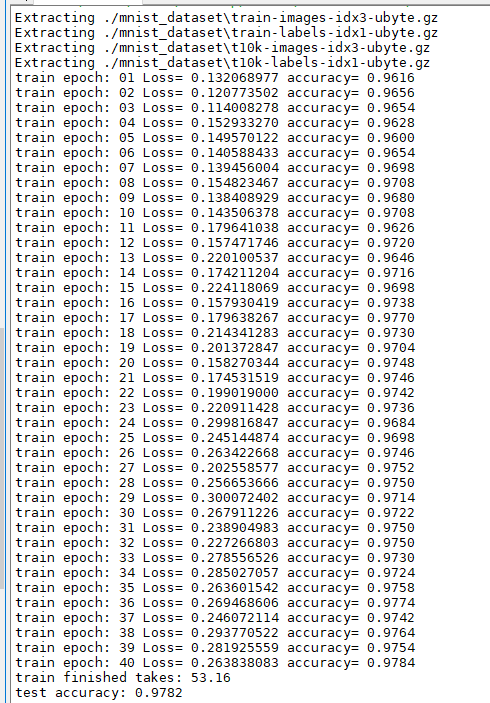




花费时间更长，训练集的准确率看起来是提高了一丢丢，但测试集的准确率却降低了一丢丢

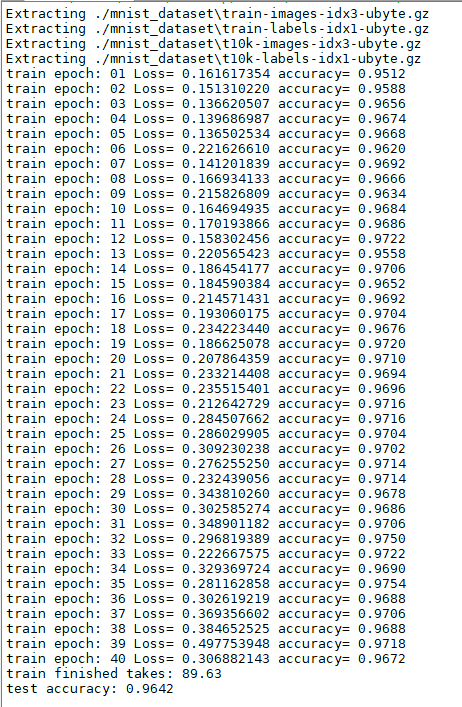
batch size :由50->100

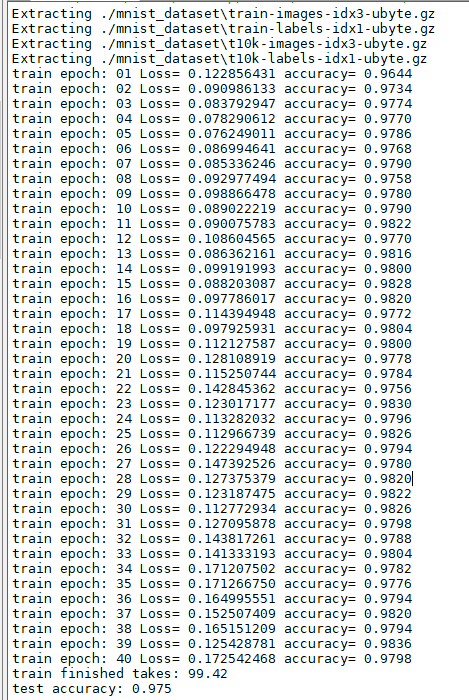




花费时间更少，准确率提高比较大

learning rate:由0.01->0.001





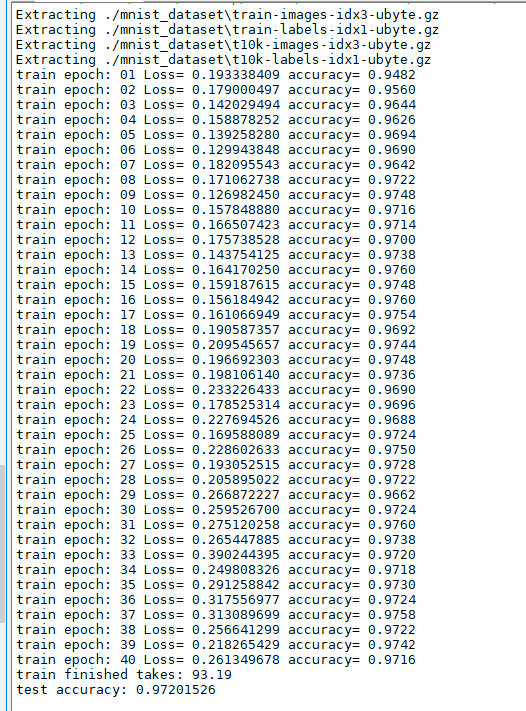
执行时间变长，准确率提高

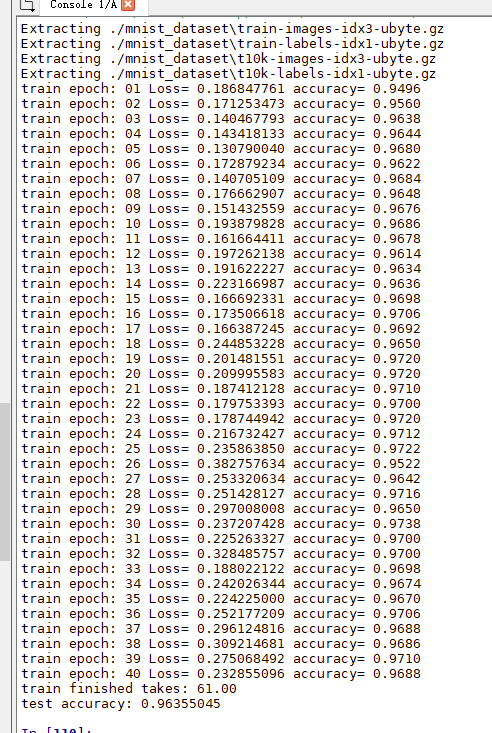
如果学习率太大，易损失函数爆炸，易震荡

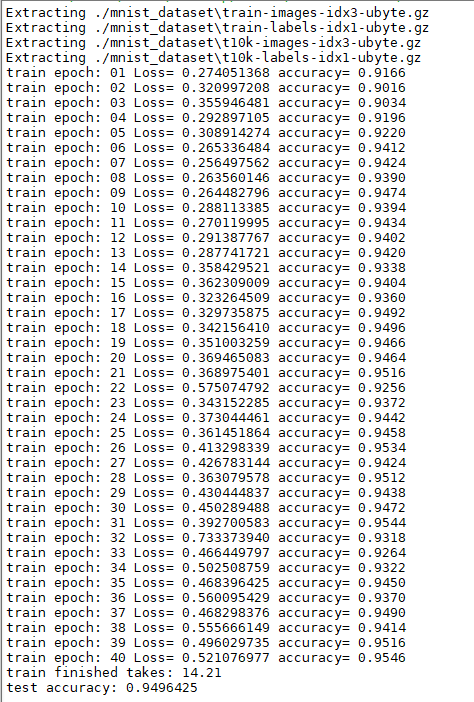
如果学习率太小，易过拟合，收敛速度慢

1. 留出法不同比例对结果的影响和分析 （10分）

训练集占比：由0.9->0.6->0.1



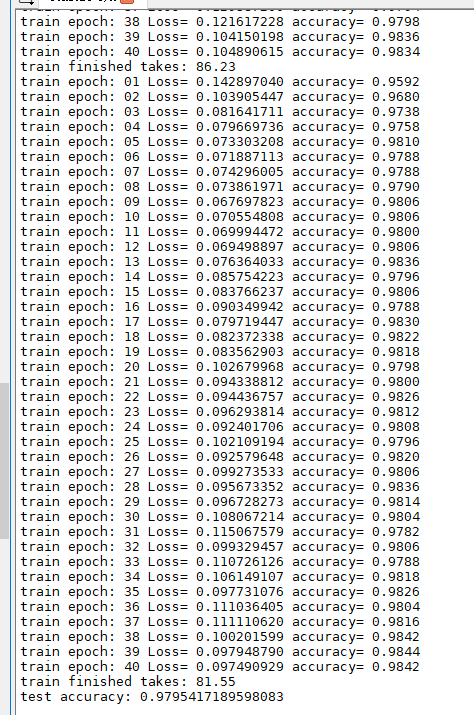


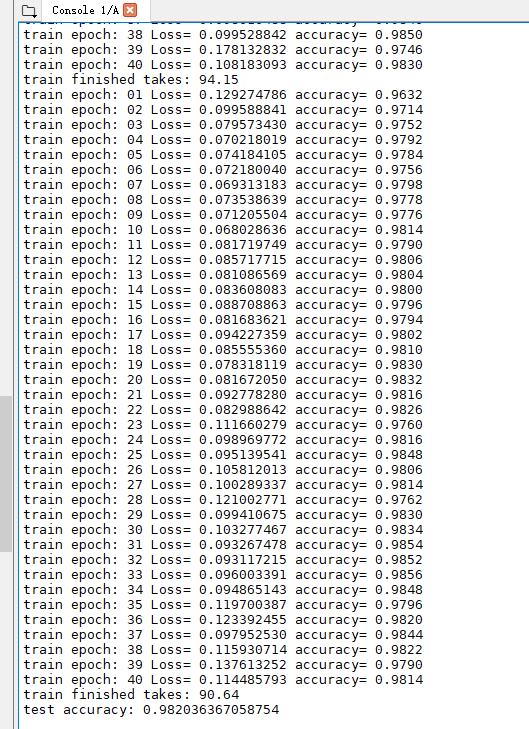


准确率下降，因为训练集越少，训练出的参数对于测试集的会不太适应

1. k折交叉验证法不同k值对结果的影响和分析 （10分）

k的取值：由5->20





准确率上升。K越大，训练的模型会更加准确，更加稳定

**五、实验总结及心得**

**这次试验首先学习配环境，因为2.0对于1.0的某些代码运行不了，也有些模块不存在。**

<https://blog.csdn.net/qq_43060552/article/details/103189040>

<https://stackoverflow.com/questions/37383812/tensorflow-module-object-has-no-attribute-placeholder>

**其次学习不同超参数对于模型训练和应用的影响，而对于不是超参数的参数，只能通过训练来优化，例如神经网络中的权重。为了让模型更好的进行应用，需要准确率达到可以是我们满意的程度，就需要人为的不断调整超参数的设置。本次主要学习的是对于机器学习中的模型的训练和应用，还学会了对hold on和k-fold的算法的编写。最后在编写的过程中遇到n多程序代码的问题，锻炼了写代码和debug能力。程序可见附件**