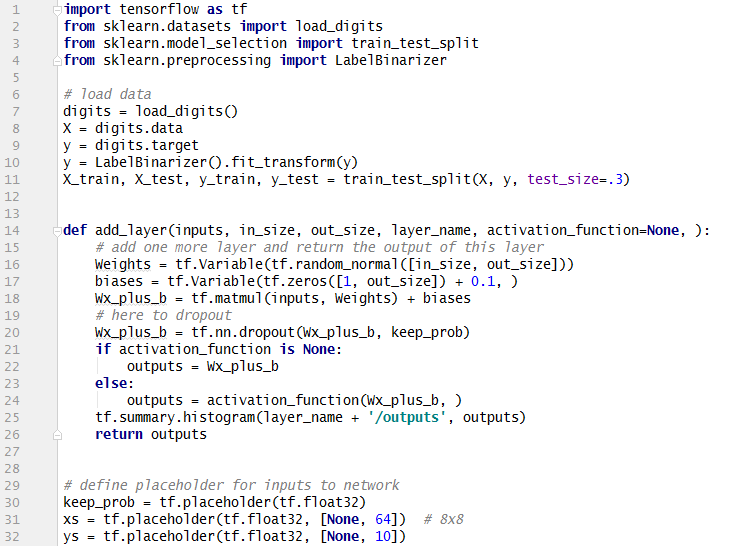
# Tensorflow相关学习

## 一、用dropout方法解决overfitting问题

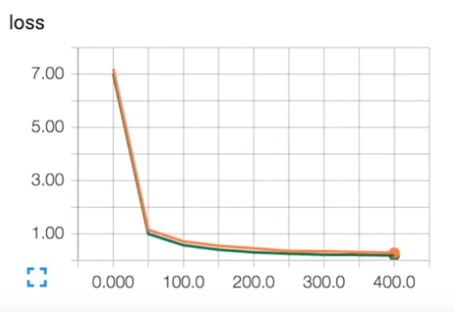
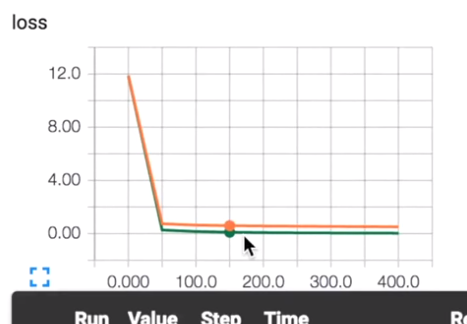
这部分内容运用到了前面学习解决过拟合方法中的dropout方法，在构建网络结构时沿用了之前已经构建好的神经网络层，并且在运行时用到了之前学习的tensorboard。

由于dropout方法是在输出时随机舍弃一部分结果，在此次用到的代码中，关键的一个步骤就是定义keep\_prob = tf.placeholder(tf.float32)。keep\_prob即为保留的概率，如果想舍弃30%的输出结果，那么keep\_prob应设置为0.7。详细代码截图如下：





成功运行后，可以发现D盘logs文件夹中出现了train和test两个子目录，里面生成了此次的日志文件。然后运行tensorboard：



如图，左边的是未经处理的loss值在training data和test data运行时的变化过程，可以发现两条曲线是有一定差值的；右边是经过drop一半的output，即进行dropout处理过的loss值变化过程，可以发现两条曲线比之前更加吻合，overfitting得到了解决。

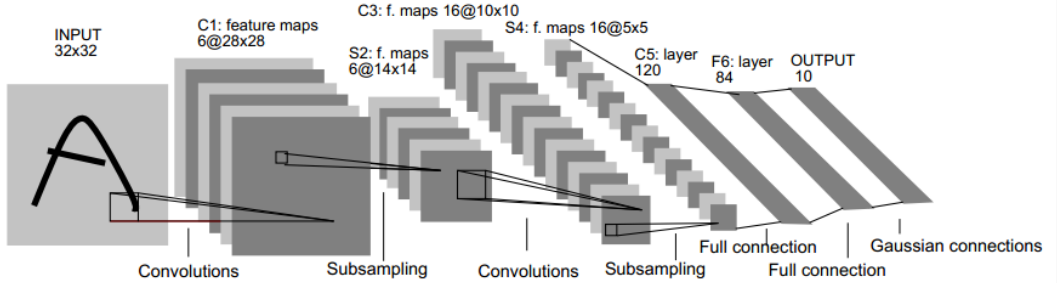
## 二、初识和简单实现CNN(Convolutional Neural Networks)

卷积神经网络就是由卷积层、池化层、全连接层构成的具有局部感知和权值共享能力的深层神经网络。它最主要的特点就是局部感知和权值共享。局部感知使其每次只需感知较小的区域，降低了参数数量，也提供了特征拟合能力（特征简单了，拟合更容易）。而全值共享，使一些基本特征可得到重复利用，使参数可以共享，提高了神经网络的训练效果。

**CNN的网络结构**：

对于CNN，它主要有两种类型的网络层，分别是卷积层和池化/采样层(Pooling)。卷积层是卷积核在上一级输入层上通过逐一滑动窗口计算而得，卷积核中的每一个参数都相当于传统神经网络中的权值参数，与对应的局部像素相连接，将卷积核的各个参数与对应的局部像素值相乘之和，（通常还要再加上一个偏置参数），得到卷积层上的结果。它的作用是提取图像的各种特征；池化层的作用是对原始特征信号进行抽象，从而大幅度减少训练参数，另外还可以减轻模型过拟合的程度，通常有两种方法：Max-Pooling: 选择Pooling窗口中的最大值作为采样值；Mean-Pooling: 将Pooling窗口中的所有值相加取平均，以平均值作为采样值。

下图是一个经典的CNN结构，称为LeNet-5网络。LeNet-5共有7层（不包含输入），每层都包含可训练参数。



①输入层

输入图像大小为32\*32，比MNIST数据集的图片要大一些，这么做的原因是希望潜在的明显特征如笔画断点或角能够出现在最高层特征检测范围的中心。因此在训练整个网络之前，需要对28\*28的图像加上paddings（即周围填充）。

②C1层

该层是一个卷积层。使用6个大小为5\*5的卷积核对输入层进行卷积运算，特征图尺寸为32-5+1=28，因此产生6个大小为28\*28的特征图。这么做够防止原图像输入的信息掉到卷积核边界之外。

③S2层

该层是一个池化层（pooling，也称为下采样层）。这里采用max\_pool（最大池化），池化的size定为2\*2，经池化后得到6个14\*14的特征图，作为下一层神经元的输入。

④C3层

该层仍为一个卷积层，我们选用大小为5\*5的16种不同的卷积核。这里需要注意C3中的每个特征图，都是S2中的所有6个或其中几个特征图进行加权组合得到的。输出为16个10\*10的特征图。

⑤S4层

该层仍为一个池化层，size为2\*2，仍采用max\_pool。最后输出16个5\*5的特征图，神经元个数也减少至16\*5\*5=400。

⑥C5层

该层继续用5\*5的卷积核对S4层的输出进行卷积，卷积核数量增加至120。这样C5层的输出图片大小为5-5+1=1。最终输出120个1\*1的特征图。这里实际上是与S4全连接了，但仍将其标为卷积层，原因是如果LeNet-5的输入图片尺寸变大，其他保持不变，那该层特征图的维数也会大于1\*1。

⑦F6层

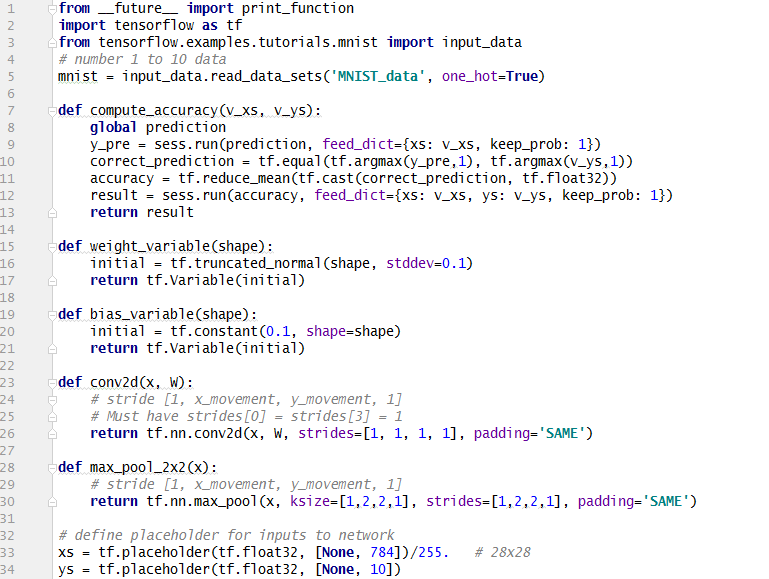
该层与C5层全连接，输出84张特征图。

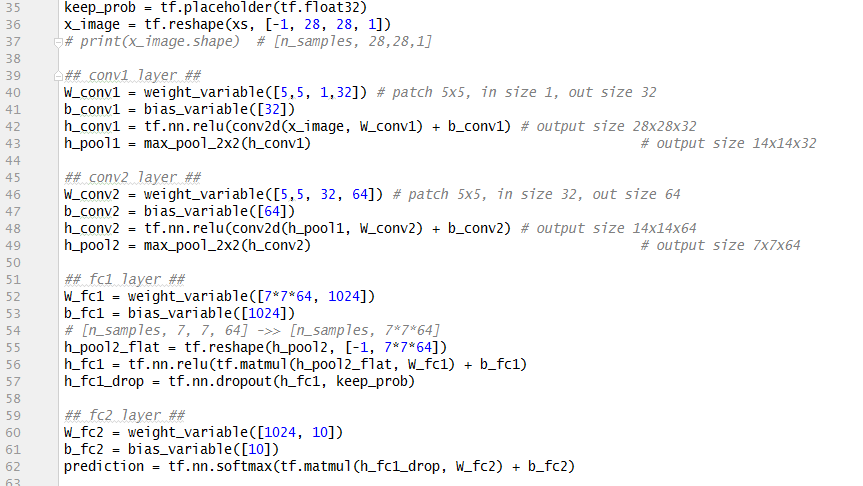
⑧输出层

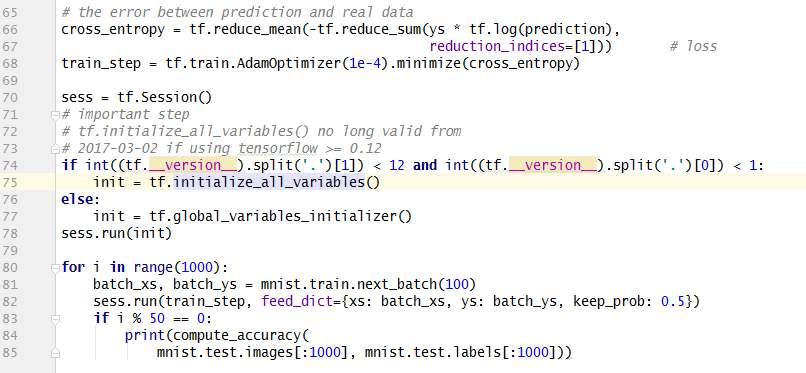
该层与F6层全连接，输出长度为10的张量，代表所抽取的特征属于哪个类别。（例如[0,0,0,1,0,0,0,0,0,0]的张量，1在index=3的位置，故该张量代表的图片属于第三类）。

**CNN的简单实现**

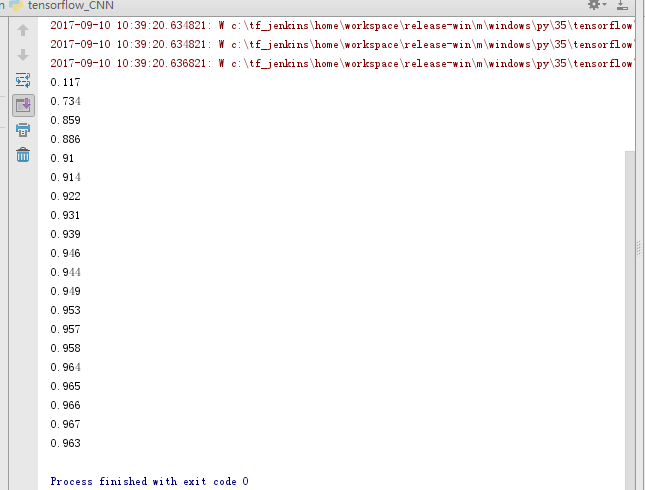
下面展示莫烦视频教学中的一个例子，用python代码实现一个卷积神经网络，对MNIST数据集中的图片进行预测并显示正确率。







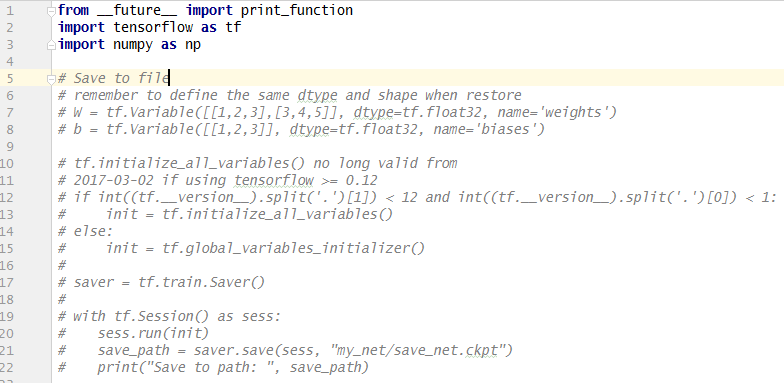
运行成功后会每隔50步打印一次正确率：



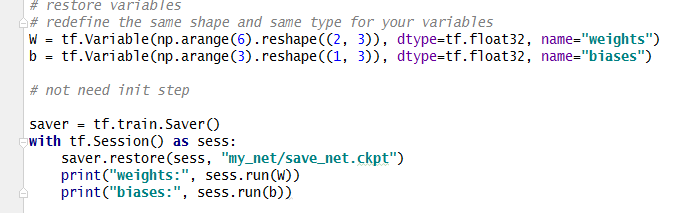
## 三、神经网络的保存和提取

对于已经构建好的神经网络，再某一次后续工作中如果想调用的话，此时需要实现的就是神经网络的保存和提取功能。代码如下：

对于保存功能：



提取功能：



目前在进行对已保存的神经网络进行提取时，仍需要先手动创建相同的网络结构，最终实现提取到的内容是原神经网络的各项参数。