

深度学习实验报告

基于tensorflow的手写数字识别

学 院： 电子信息工程学院

专 业： 通信工程

姓 名： 刘美鹭

学 号： 17211296

班 级： 通信1701班

**北京交通大学**

2020年5月

**目 录**

[一、实验目的 - 1 -](#_Toc41155167)

[二、实验环境 - 1 -](#_Toc41155168)

[三、实验原理 - 1 -](#_Toc41155169)

[**3.1制作并收集数据集** - 1 -](#_Toc41155170)

[**3.2配置系统环境** - 1 -](#_Toc41155171)

[**3.3导入MNIST数据集，做好标签和分类** - 2 -](#_Toc41155172)

[**3.4数据格式** - 3 -](#_Toc41155173)

[**（一）数据存储格式** - 3 -](#_Toc41155174)

[**（二）数据存储方式** - 3 -](#_Toc41155175)

[**（三）输出图像数据** - 4 -](#_Toc41155176)

[**3.5数据预处理** - 4 -](#_Toc41155179)

[**3.6建立模型及设置图层** - 5 -](#_Toc41155180)

[**3.7编译模型** - 5 -](#_Toc41155181)

[**（一）回归损失：** - 5 -](#_Toc41155182)

[**（二）分类损失：** - 6 -](#_Toc41155185)

[**3.8训练模型并喂模型** - 7 -](#_Toc41155188)

[**3.9评估准确性并作出预测** - 8 -](#_Toc41155189)

[四、创建模型 - 8 -](#_Toc41155190)

[**4.1导入数据集** - 8 -](#_Toc41155191)

[**4.2创建模型** - 8 -](#_Toc41155192)

[**4.3搭建网络：** - 9 -](#_Toc41155193)

[**（一）第一次卷积和池化** - 9 -](#_Toc41155194)

[**（二）第二次卷积和池化** - 10 -](#_Toc41155197)

[**4.4函数拟合** - 11 -](#_Toc41155198)

[**（一）softmax函数拟合分布以及dropout函数预防过拟合方法** - 11 -](#_Toc41155199)

[**（二）定义损失和优化** - 11 -](#_Toc41155201)

[**4.5模型评价** - 12 -](#_Toc41155202)

[五、预测模型 - 12 -](#_Toc41155203)

[六、实验结果 - 14 -](#_Toc41155204)

[七、实验体会 - 16 -](#_Toc41155205)

一、实验目的

1.通过学习深度学习的内容更加了解当今世界的发展趋势。

2.通过深度学习熟练掌握python编程的过程，对配置环境和python等语言有更深入的了解。

3.熟悉当今机器学习的大环境下，使用Google提供的tensorflow环境来跑代码，熟悉基本的深度学习方法。

二、实验环境

Windows10 python3.6 tensorflow1.14.0

1. 实验原理

**3.1制作并收集数据集**

制作和收集数据集是深度学习的首要步骤。这通常不是一件容易的事，需要投入时间和金钱。将结果与不同的数据集大小进行比较，并尝试进行推断。MNIST是最受欢迎的深度学习数据集之一。这是一个手写数字数据集，包含一组60,000个示例的训练集和一组10,000个示例的测试集。这是一个对于在实际数据中尝试学习技术和深度识别模式的很好的数据库，同时尝试学习如何在数据预处理中花费最少的时间和精力。

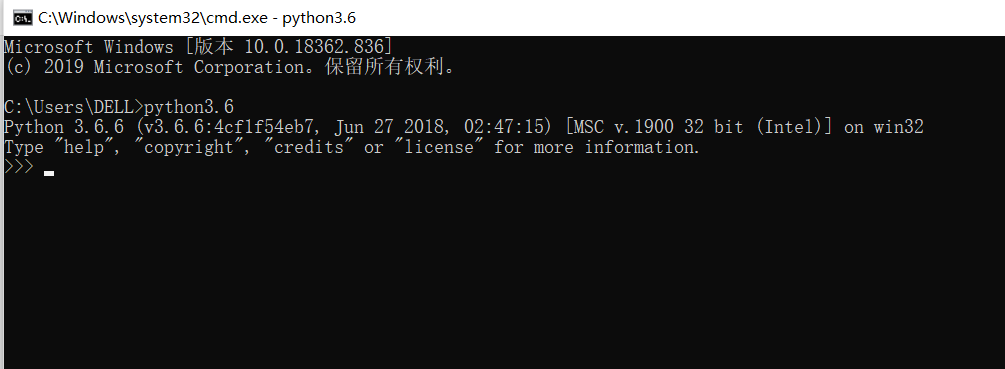
**3.2配置系统环境**

图3-1 安装python3.6

首先在官网<https://www.python.org/>上下载python3.6版本，下载之后在命令行打开应该显示上图信息

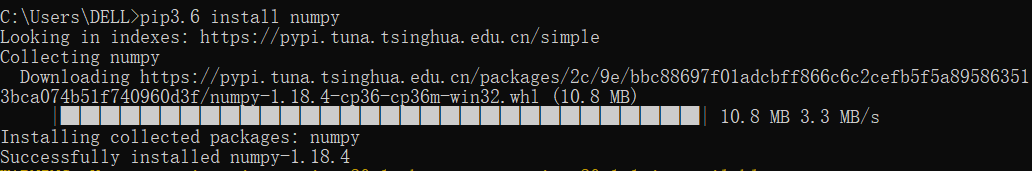
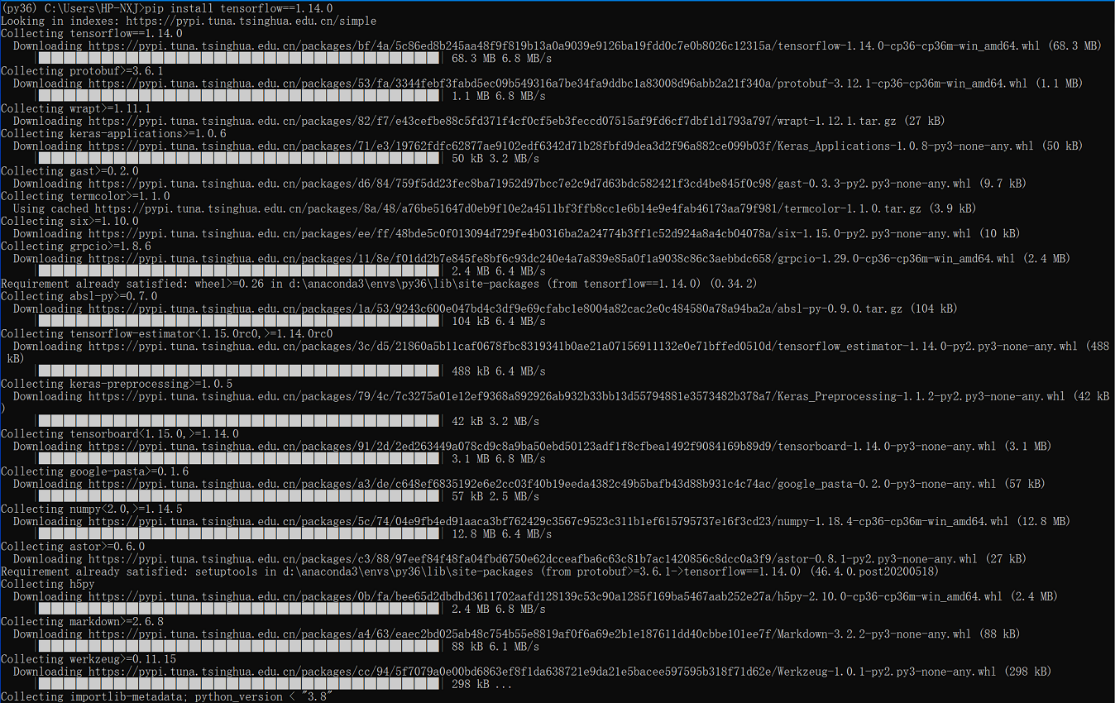
并安装pip3.6添加环境变量并使用pip下载numpy包，如下图所示

图3-2 安装pip以及numpy包

紧接着进入清华镜像网站https://mirror.tuna.tsinghua.edu.cn/ tensorflow/ 下载tensorflow1.14.0配置好环境后就可以运行代码了。

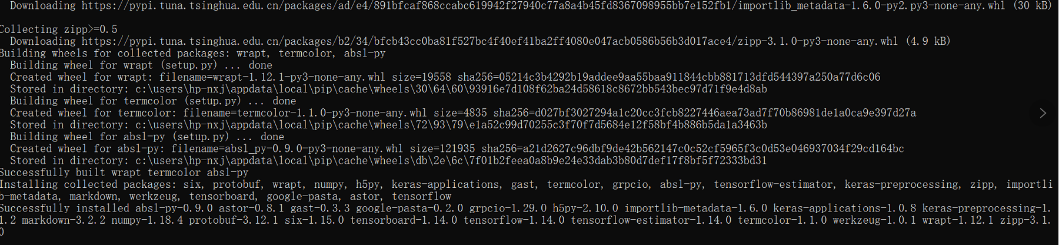
图3-3 安装(1)tensorflow

图3-4 安装(2)tensorflow

**3.3导入MNIST数据集，做好标签和分类**

每个图像都映射到一个标签。由于类名不包含在数据集中，因此将它们存储在此处以供以后在绘制图像时使用。

**3.4数据格式**

**（一）数据存储格式**

在训练模型之前，需要探索数据集的格式。下图显示了训练集中有60,000张图像，每张图像表示为28 x 28像素。

另一个重要的数据格式就是通道，是数字图像中存储不同类型信息的灰度图像。一个图像最多可以有数十个通道，常用的RGB和Lab图像默认有三个通道，而CMYK图像则默认有四个通道。一张RGB图像含有三个通道：红（Red）、绿（Green）、蓝（Blue）。一张CMYK图像含有四个通道：青色（Cyan）、品红（Magenta）、黄色、黑色。

所以想灰度图就只有一个通道，占有8个bit位，也就是8位图。所以RGB图像占有三个通道，3\*8=24，所以RGB图像就是24位图。

**（二）数据存储方式**

图像像素点的存储就是对应的原图从左到右，从上到下，依次排列，每个点的值就是像素点的值，每个点的地址就是像素点的地址。

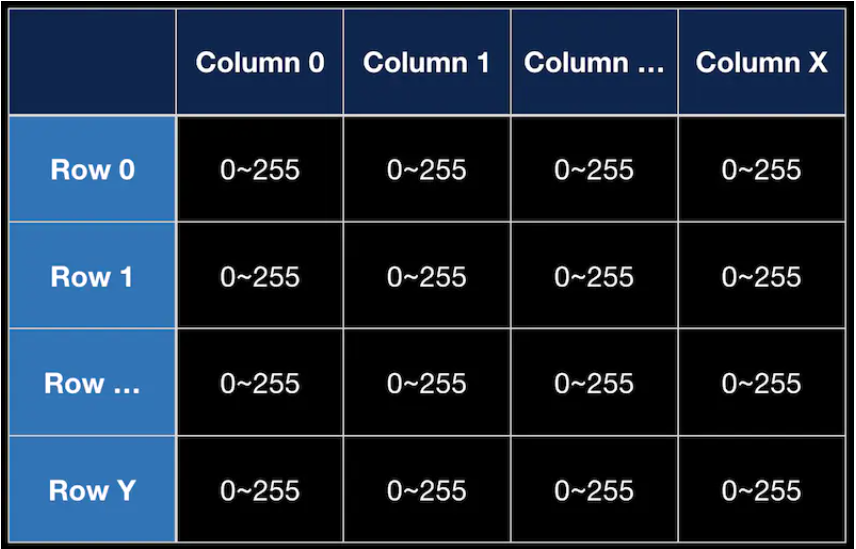
如图3-5就是灰度图的存储，只有单通道。在内存中的存储即可用一个一维数组来表示，根据顺序从左到右，从上到下，依次按顺序存入数组。

图3-5 灰度存储数据

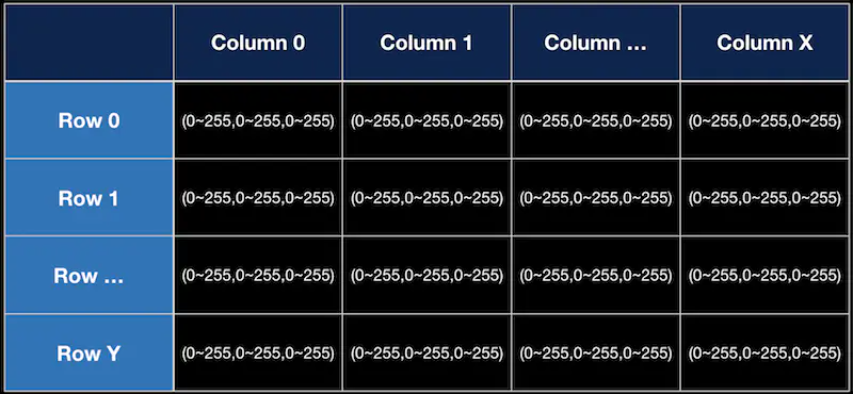
如图3-6则为RGB图像的存储模型，每一个像素有3个通道，所以需要一个二维数组来表示，顺序也是从左到右，从上到下，如[[234,200,0],[234,0,0],[255,55,0],....]这样，当然其中的数字，在内存中需要用对应的二进制来表示。

图3-6 RGB存储数据

**（三）输出图像数据**

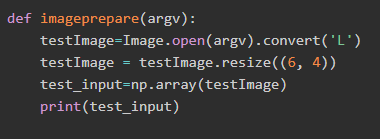
可以选择传进去一张图片，然后转换成L模型（L表示灰度图），设置宽高位（6，4），输出如下所示，这里np.array把图片数据转换成了一个二维数组，方便根据（x,y）来读取，一般可以采取如下代码来输出：

图3-7 输出图像数据

**3.5数据预处理**

在训练网络之前，必须对数据进行预处理。一般为使得像素值落在0到255之间，会用如下代码对于图像进行处理。

train\_images = train\_images / 255.0  
test\_images = test\_images / 255.0

在将它们输入神经网络模型之前，将这些值缩放到0到1的范围。为此，将值除以255。以相同的方式预处理训练集和测试集非常重要。

**3.6建立模型及设置图层**

建立神经网络需要配置模型的各层，然后编译模型。经网络的基本构建模块是层。图层从输入到其中的数据中提取表示。希望这些表示对于当前的问题有意义。深度学习的大部分内容是将简单的层链接在一起。大多数图层（例如tf.keras.layers.Dense）具有在训练期间学习的参数。

在本实验中针对手写数字识别：

首先，网络的第一层将图像的格式从二维数组（28 x 28像素）转换为一维数组（28 \* 28 = 784像素）。可以将这一层看作是堆叠图像中的像素行并对齐它们。该层没有学习参数。它只会重新格式化数据。

其次，在像素展平后，网络由两个层序列组成。 这些是紧密连接或完全连接的神经层。 第一密集层具有128个节点（或神经元）。 第二层（也是最后一层）返回长度为10的logits数组。每个节点包含一个得分，该得分指示当前图像属于10类之一。

**3.7编译模型**

在准备训练模型之前，需要进行一些其他设置。 这些是在模型的编译步骤中添加的，损失函数分为了两大类：分类和回归。

然后又分别对这两类进行了细分和讲解，其中回归中包含了一种不太常见的损失函数：平均偏差误差，可以用来确定模型中存在正偏差还是负偏差。损失函数-衡量训练期间模型的准确性。

**（一）回归损失：**

回归损失大致分为两类：

第一种方式：均方误差（MSE）度量的是预测值和实际观测值间差的平方的均值。它只考虑误差的平均大小，不考虑其方向。但由于经过平方，与真实值偏离较多的预测值会比偏离较少的预测值受到更为严重的惩罚。再加上 MSE 的数学特性很好，这使得计算梯度变得更容易。一般使用如下代码：

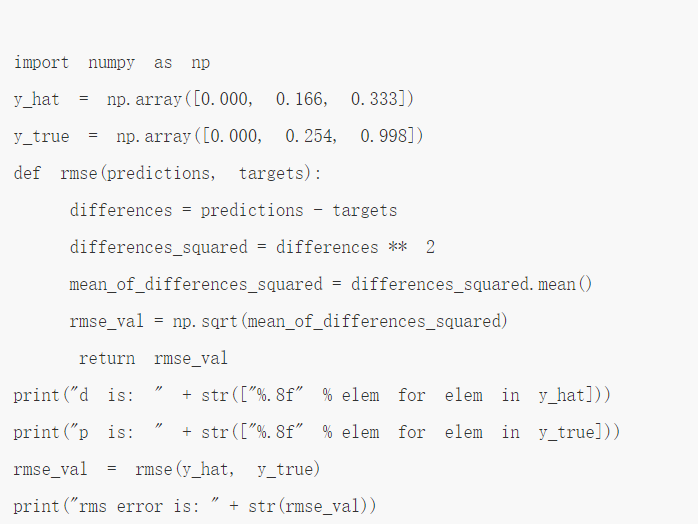


图3-8 均方误差（MSE）度量

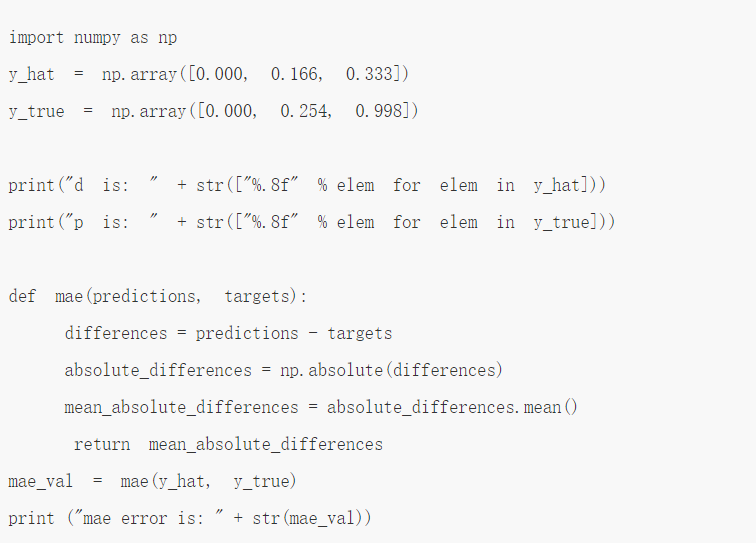
第二种方式：平均绝对误差（MAE）度量的是预测值和实际观测值之间绝对差之和的平均值。和 MSE一样，这种度量方法也是在不考虑方向的情况下衡量误差大小。但和 MSE 的不同之处在于，MAE 需要像线性规划这样更复杂的工具来计算梯度。此外，MAE 对异常值更加稳健，因为它不使用平方。

图3-9 平均绝对误差（MAE）度量

**（二）分类损失：**

交叉熵损失/负对数似然是分类问题中最常见的设置。随着预测概率偏离实际标签，交叉熵损失会逐渐增加。公示如下

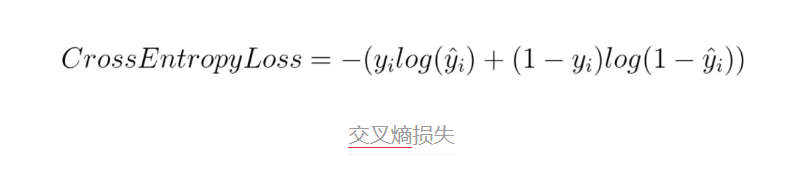


图3-9 交叉熵损失公示

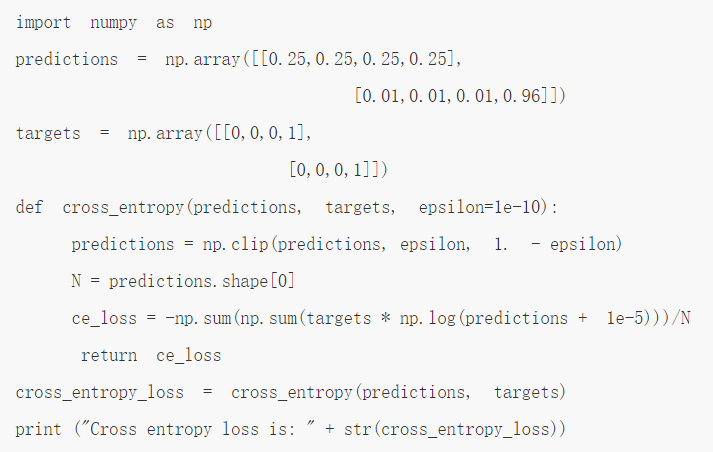
注意，当实际标签为 1即(y(i)=1) 时，函数的后半部分消失，而当实际标签是为 0即(y(i)=0) 时，函数的前半部分消失。简而言之，只是把对真实值类别的实际预测概率的对数相乘。还有重要的一点是，交叉熵损失会重重惩罚那些置信度高但是错误的预测值。一般使用如下代码完成：

图3-9 交叉熵方法

**3.8训练模型并喂模型**

训练神经网络模型需要执行以下步骤：将训练数据输入模型。

可以把训练数据放在train\_images和train\_labels数组中。该模型学习关联图像和标签。要求模型对测试集进行预测。验证预测是否与test\_labels数组中的标签匹配。开始训练，可以调用model.fit方法，之所以这么称呼是因为它使模型“适合”训练数据。

**3.9评估准确性并作出预测**

事实证明，测试数据集的准确性略低于训练数据集的准确性。 训练准确性和测试准确性之间的差距代表过度拟合。当机器学习模型在新的，以前看不见的输入上的表现比训练数据上的表现差时，就会发生过度拟合。 过度拟合的模型“记忆”训练数据集中的噪声和细节，从而对新数据的模型性能产生负面影响，并预测测试集中每个图像的标签。所谓过拟合即为过分分析原图像，将应该准确识别的东西进行了误判。

四、创建模型

主要的创建模型的过程在model.py中，使用MNIST数据集训练并保存模型。

**4.1导入数据集**

在代码的模型中，导入MNSIT数据集,运行时会自动检查运行的时候是否已经有数据集了，如果没有会自动下载MNIST数据集。

图4-1 加载数据集

**4.2创建模型**

首先要用placeholder函数占位，并定义函数W和函数b，并给出推算y的网络结构。x代表的是输入图片的浮点数张量，因此定义dtype为"float"。其中，shape的None代表了没有指定张量的shape，可以feed任何shape的张量，在这里指batch的大小未定。一张mnist图像的大小是28\*28，784是一张展平的mnist图像的维度，即28\*28＝784。如图4-2。

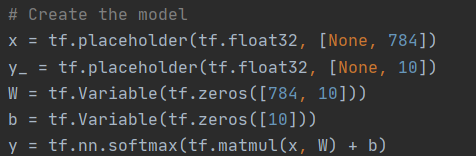


图4-2 创建模型

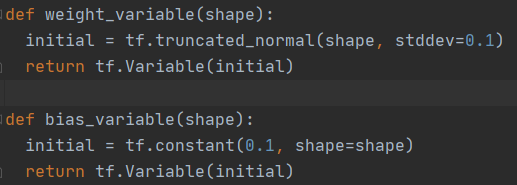
有参数W用weight\_variable函数实现 参数b用bias\_variable函数实现。

图4-3 实现参数W和参数b

**4.3搭建网络：**

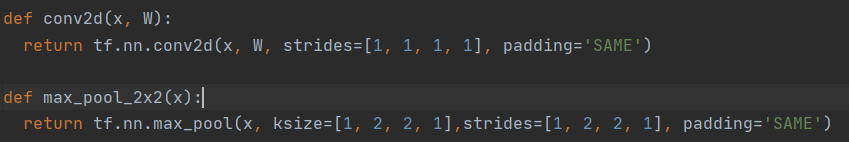
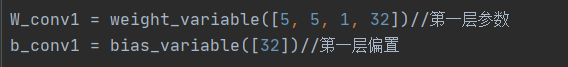
**（一）第一次卷积和池化**

图4-4 定义卷积和池化

搭建网络的过程即，描述从输入到输出的数据流。在这里使用步长为相同填充(padding='SAME')的办法进行卷积。与此同时，使用2x2的网格以max pooling的方法池化。

图4-5 设置第一层卷积和偏置

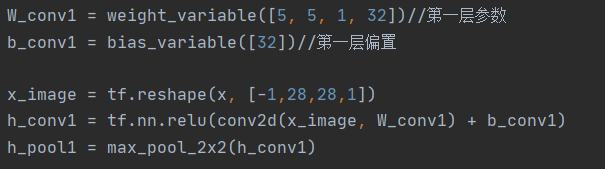
第一层卷积W\_conv1表示在5\*5的像素点中取32特征图，也即卷积核个数，输出的通道数为32，所以偏置项b\_conv1也为32。

图4-6 第一次卷积和池化

Reshape函数将其转换为四维向量，其中第一个参数－1是指可以先不指定，第二和第三个参数是指图像的大小，第四个参数对应颜色通道数目，灰度图对应1，rgb图对应3，本实验中的示例代码是先将图片转换为灰色。

函数中权重与偏置项初始化过程中使用RELU函数，而目前使用较多的就是relu函数。它模拟了生物学上的阈值响应机制，利用人脑只对大于某个值的信号才产生响应的机制，提出了单侧抑制的理念。它的表达式很简单，f(x)=max(0,x)。当x>0时，y=x, x<0时，y=0.其函数图像如下图4-7所示

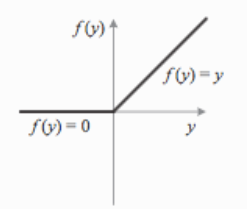


图4-7 激活函数RELU图像

由上面的图像可知，在x<0的时候，随着训练的进行，很有可能进入饱和区，导致权重无法更新，所以在代码中加入了一个偏移量b\_conv1，避免训练函数陷入死循环。

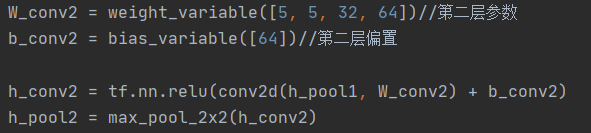
**（二）第二次卷积和池化**

图4-8 第二次卷积和池化

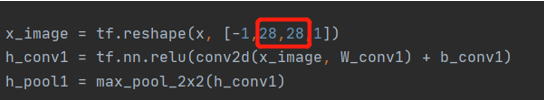
第二次卷积和池化的过程与第一次十分相似，每一次的卷积池化都会让图片的长和宽减半，初始值为28，经过两次卷积池化最后变为7\*7的像素点。

图4-9 选取图片初始值

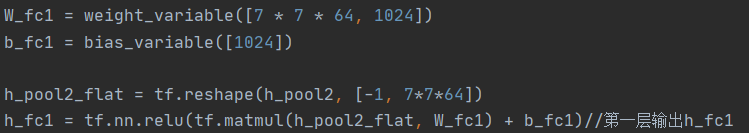
在加入一个1024个神经元的全连接后，再次使用reshape函数转换，并使用RELU函数激活。

图4-10 转换并激活

**4.4函数拟合**

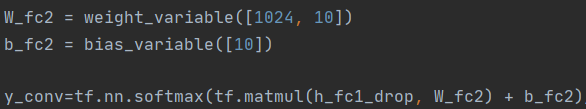
**（一）softmax函数拟合分布以及dropout函数预防过拟合方法**

图4-11 softmax函数拟合分布

第二层输出tf.matmul(h\_fc1\_drop, W\_fc2) + b\_fc2,对其使用softmax函数使其满足概率分布函数。

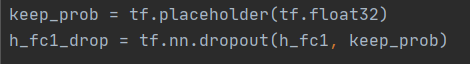


图4-12 dropout方法防止过拟合

dropout方法：该方法非常简单，每次完全随机选取神经网络一半的数据来训练，把被抛弃的激活值用零来替代。

**（二）定义损失和优化**

计算交叉熵的代价函数，并使用优化算法使得代价函数最小化，再找出预测正确的标签，Correct\_prediction返回1维度y\_中最大值所在的索引号，y是神经网络喂入的banchsize\*10的一维数组，1表示选取最大值的操作仅在第一个维度进行返回每一行最大值所对应的列表索引号，会得到一个长度为banch的一维数组，每一轮样本推算出的识别结果，equal判断两个张量中每一维是否相等，相等返回true,否则返回fause。

再将得出通过正确个数除以总数得出准确率，将x转换为tf.float32格式，将bool类型转换为实数类型，然后计算平均值。具体实现如下图所示:

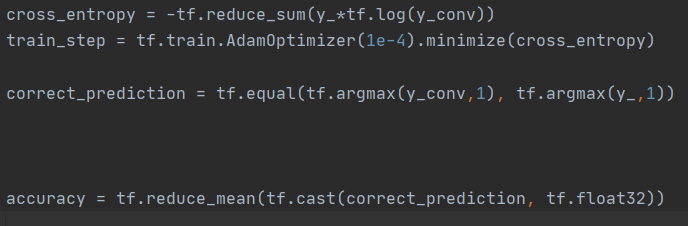


图4-13 损失和优化

**4.5模型评价**

每个一定轮数，都要保存一遍模型，实例化saver对象。在with结构中当for循环重复一定的轮数的时候，保存模型到当前的绘画。

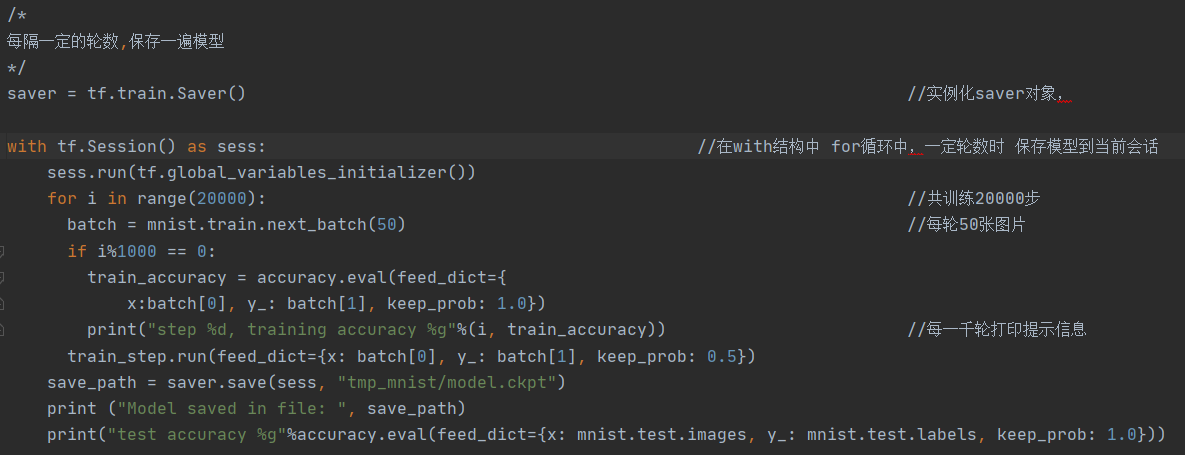
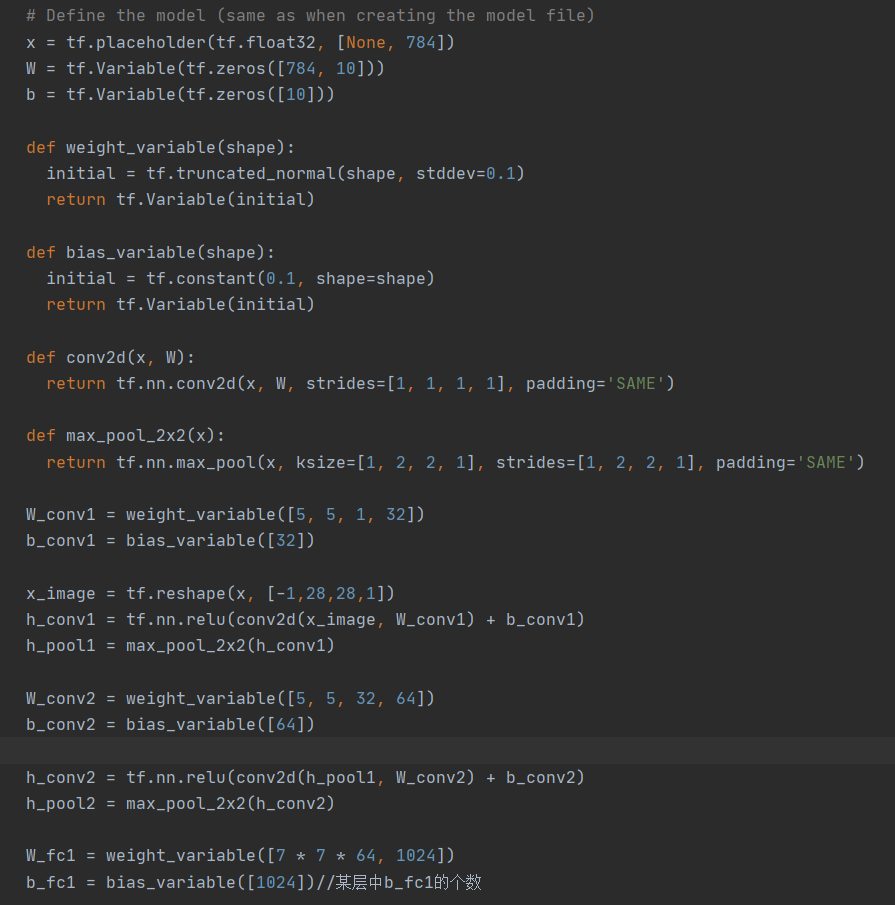
在下图给出的代码中，共训练20000步，每轮训练50张图片，每循环1000轮打印提示信息。

图4-14 模型评价

五、预测模型

解析图片并预测图片主要在predict1.py中，解析手写数字图片并调用模型文件识别图片中的数字。

首先要想创建模型一样的定义一个模型，方法类似于第四部分，这里不再赘述，代码实现过程为如下图片。

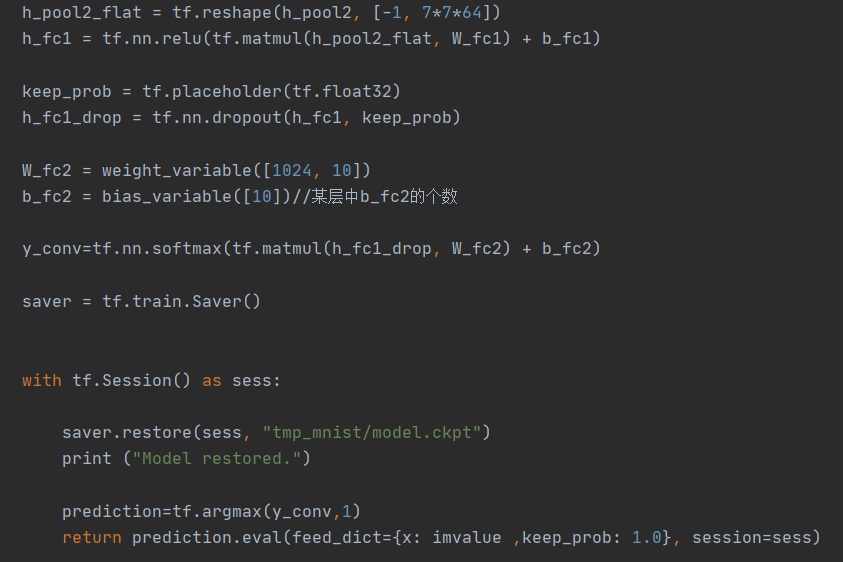
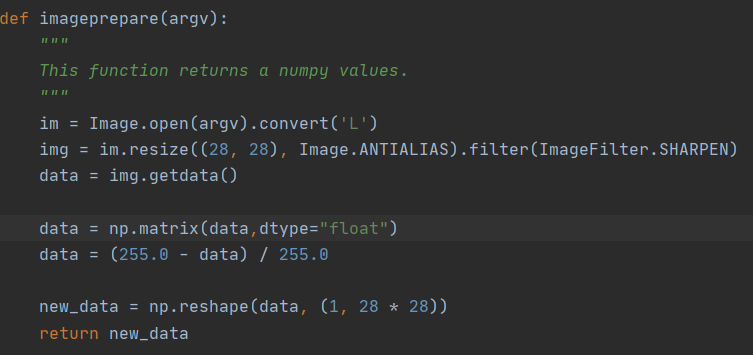
图5-1 创建模型1

图5-2 创建模型2

其次要显示模型，并选择灰度模型输出，定义长宽为28:28，获取图片的数字信息并存储在矩阵中。具体代码如下图所示

图5-3 输出图片

六、实验结果

如下图为训练过程及结果，每一千轮打印一次结果和精度。

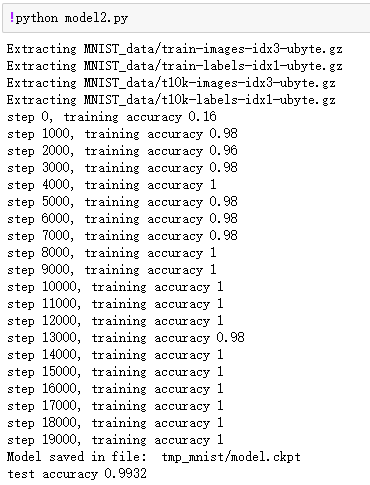


图6-1 测试训练结果和精度

下图6-2为测试集图片：

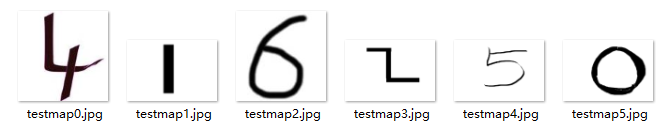


图6-2 测试训练结果和精度

如下几张图为每张图片的测试结果：

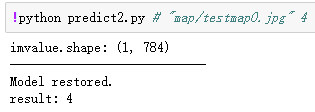


图6-3 测试结果1

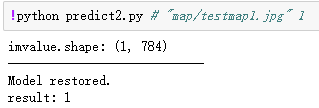


图6-4 测试结果2

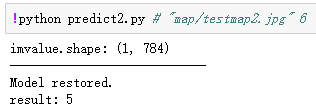


图6-5 测试结果3

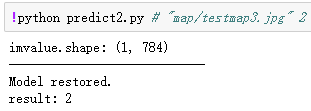


图6-6 测试结果4

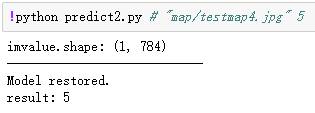


图6-7 测试结果5

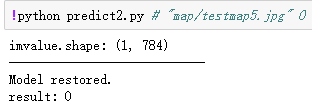


图6-8 测试结果6

在所有测试结束后发现，测试样例预测良好，其中仅有测试集testmap2出错。

七、实验体会

在学习了深度学习的基本知识后，已经对深度学习产生了浓厚的兴趣，无论是深度学习的算法还是各种训练模型。但是真正写出自己的代码，做出第一个模型，就感觉其实整个创建模型，训练模型并适当调参的过程，远没有想象中的那么简单。

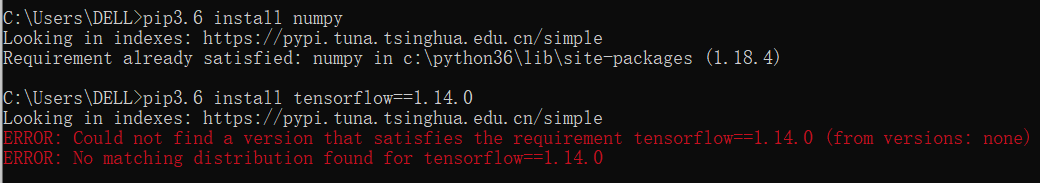
第一个困难对我来说就是环境的配置，其实花费了很长的时间去设置。首先是下载python，因为版本不对的问题，先后下载了python2.7和3.8以及最终可以使用的python3.6，然后安装pip以及numpy和tensorflow的时候先后出现pip未更新，numpy与pip版本不一致，以及tensorflow和numpy不兼容等情况，最终在同学的帮助和用了网上的很多不同的方法后得到了解决。

图7-1 安装过程中出现的问题

第二个困难就是对于模型的理解，对于cnn模型，我发现我并不能很好地理解卷积神经网络的设计原理，以及其实验过程。首先，我先是照着网上的示例代码训练了几个简单的识别程序，在学习代码的训练过程中，我逐渐了解并认识到了其模型的基本框架，无论是前向传播还是反向传播，每一个过程都要仔细地学习才让我最终了解到了它代码每行的大概意思，无论是神经网络的搭建还是函数的拟合，以及对于维度的升降，在学习的过程中，我感觉自己慢慢地熟悉了整个深度学习体系。

深度学习是机器学习的众多方法之一。其他方法包括决策树学习、归纳逻辑编程、聚类、强化学习和贝叶斯网络等。机器学习是一种“训练”算法的方式，目的是使机器能够向算法传送大量的数据，并允许算法进行自我调整和改进，而不是利用具有特定指令的编码软件例程来完成指定的任务。实验中，我认识到了对于整个深度学习的基本框架，也学习了代码的运行和模型的训练过程，也对这一方面有了自己的理解和看法，感受到了学习的乐趣。