****



**实验三：LeNet-5识别手写数字**

**学 院 智能与计算学部**

**专 业 网络空间安全**

**年 级 2020**

**姓 名 石子跃**

**学 号 3020244294**

**指导教师 何东晓**

**实验三：LeNet-5识别手写数字**

1. **实验目的**

完成基于LeNet-5的手写数字识别系统

1. **实验内容**

LeNet-5共有7层（不包含输入），每层都包含可训练参数。

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子感受野（receptive field）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0）。

C1层：该层是一个卷积层。使用6个大小为5\*5的卷积核，步长为1，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。

S2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。

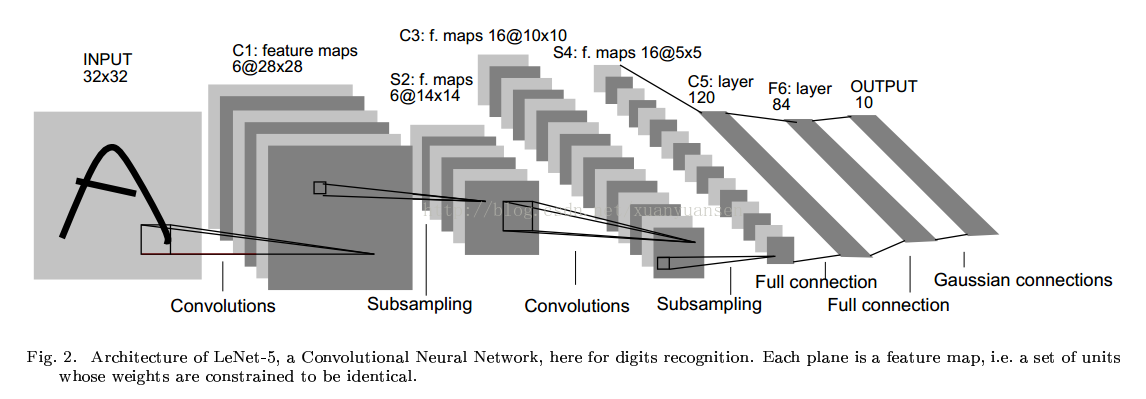
C3层：该层仍为一个卷积层，我们选用大小为5\*5的16种不同的卷积核。这里需要注意：C3中的每个特征图，都是S2中的所有6个或其中几个特征图进行加权组合得到的。输出为16个10\*10的特征图。

S4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。

F5层：该层我们继续用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。

F6层：该层与C5层全连接，输出84张特征图。

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）



1. **实验流程**

首先打印出数据集的数据结构，了解数据集

1. **from** keras.datasets **import** mnist
3. train\_data **=** mnist.load\_data()[0]
4. test\_data **=** mnist.load\_data()[1]
5. train\_X , train\_y **=** train\_data
6. test\_X, test\_y **=** test\_data
7. print(train\_X.shape, train\_y.shape)
8. print(test\_X.shape, test\_y.shape)

然后根据需要建模的模型进行输入输出格式匹配，读取数据集

1. train\_X, train\_y **=** mnist.load\_data()[0]
2. train\_X **=** train\_X.reshape(**-**1, 28, 28, 1)
3. train\_X **=** train\_X.astype('float32')
4. train\_X **/=** 255
5. train\_y **=** to\_categorical(train\_y, 10)

根据ppt所给LeNet-5资料进行建模

1. model **=** Sequential()
3. model.add(Conv2D(6, (5,5), activation**=**'relu', input\_shape**=**(28,28,1)))
4. model.add(MaxPool2D(pool\_size**=**(2,2)))
5. model.add(Conv2D(16, (5,5), activation**=**'relu'))
6. model.add(MaxPool2D(pool\_size**=**(2,2)))
7. model.add(Flatten())
8. model.add(Dense(120, activation**=**'relu'))
9. model.add(Dense(84, activation**=**'relu'))
10. model.add(Dense(10, activation**=**'softmax'))

其中每行分别对应下面的结构，只是由于keras的minst数据集所给图片大小为28，28，所以对输入进行了修改



使用训练集进行训练，并保存模型，以供以后使用

1. model.fit(train\_X, train\_y,
2. batch\_size**=**batch\_size,
3. epochs**=**epochs)
4. model.save('Minst\_model.h5')

使用测试集对所得模型进行测试，并打印出正确性

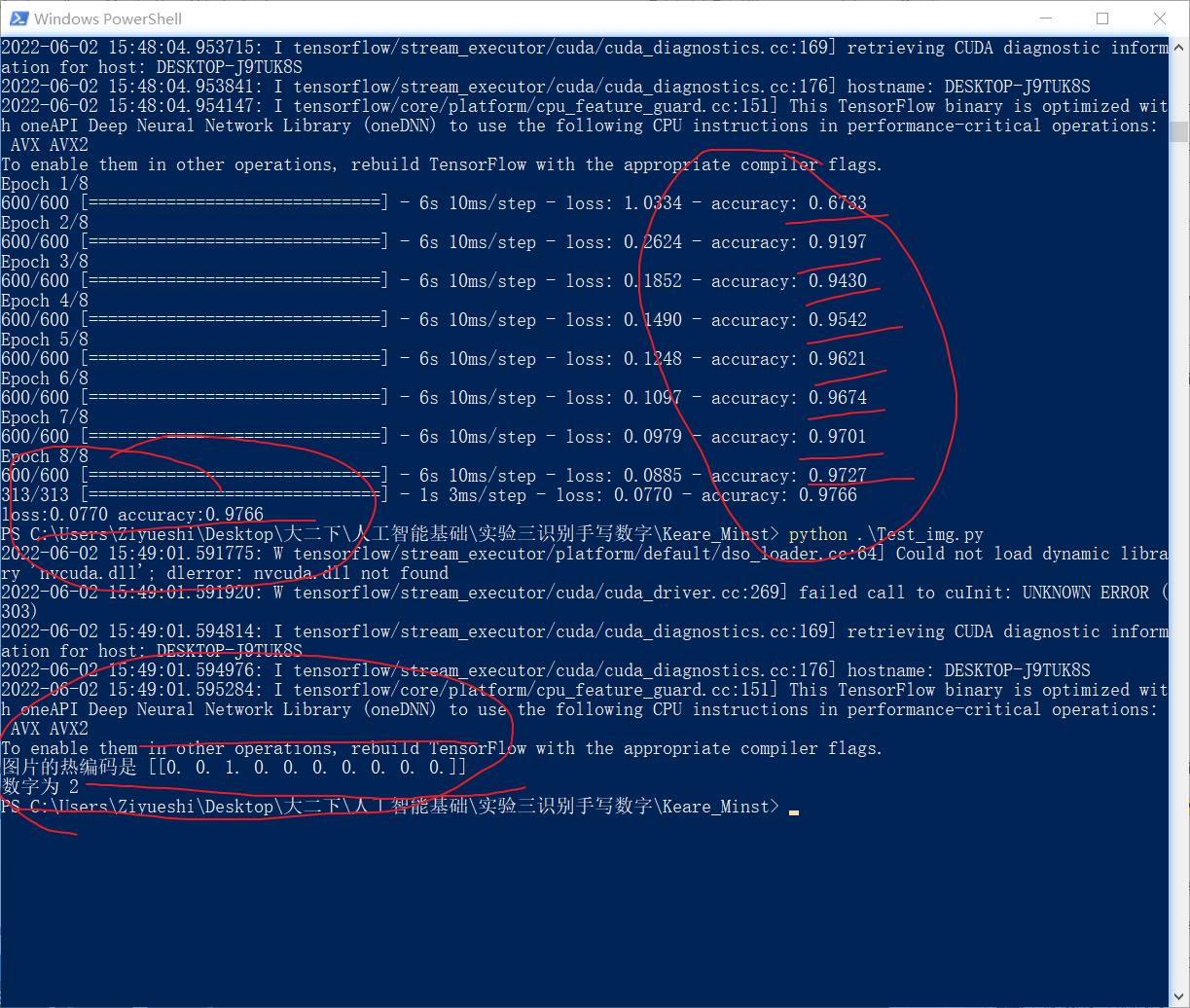
1. loss, accuracy **=** model.evaluate(test\_X, test\_y, verbose**=**1)
2. print('loss:%.4f accuracy:%.4f' **%**(loss, accuracy))
3. **实验代码**

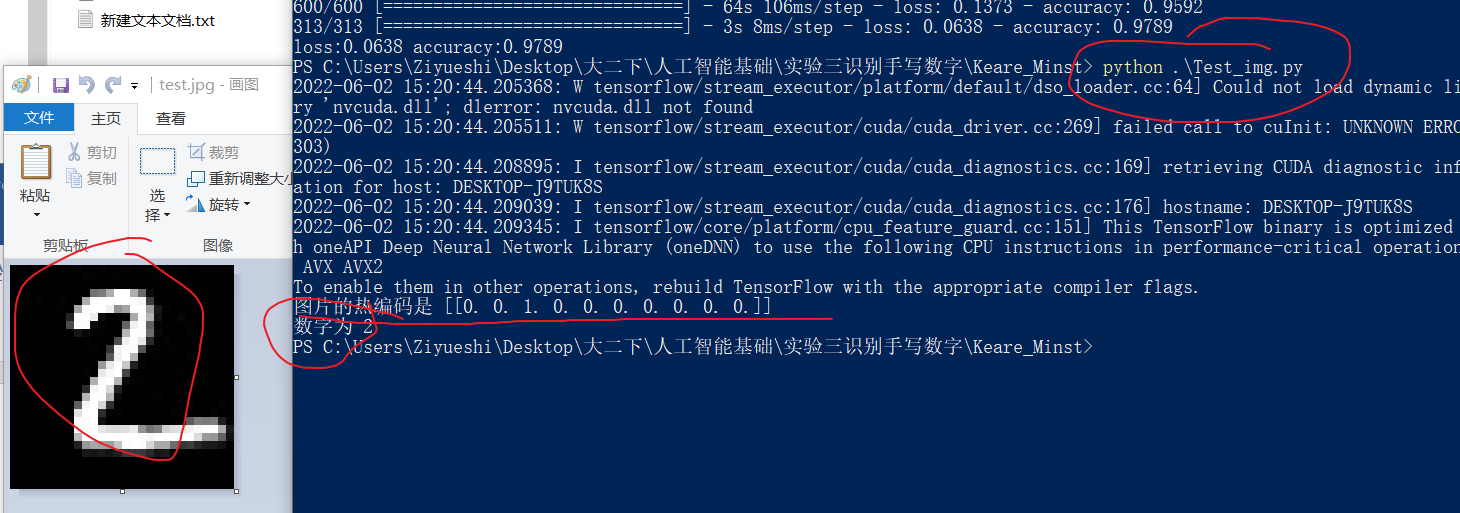
见文件中的代码

Minst.py——负责训练模型，并自动用测试集测试返回正确率，并生成.h5模型

Test\_img.py——利用.h5中的模型对自己写的图片进行预测

1. **实验结果**





最终也是正确识别了2，还可自己切换自己写的图片进行测试。

**六．实验总结**

在经历了许多弯路过后，终于找到了keras这一个机器学习库。相比于最开始我使用的tensorflow来说，keras更加简单，代码行数更少的情况下可以实现相同的效果。对于机器学习，我还有许多不清楚的需要学习的地方，未来我还会更多的接触这方面。