# Mnist识别研究

## 作业任务1：熟悉CNN与DNN（训练样本60000，测试10000）

### （0）利用CNN和DNN完成Mnist识别任务，CNN结构如下所示，DNN结构自定，将效果作为baseline。

1，使用DNN，结构

Dense(32, input\_dim=784), # 全连接层。32个神经元，输入维度为784（28\*28）  
Activation('relu'), # 激活层  
Dense(10), # 全连接层，10个神经元  
Activation('softmax') # 激活层

nb\_epoch=1, batch\_size=32

model.fit(X\_train, y\_train, nb\_epoch=100, batch\_size=32)

两次效果：

test loss: 0.21429610718488692

test accuracy: 0.9395

test loss: 0.21414068414568901

test accuracy: 0.9399

2，使用CNN结构

conv\_1 = Conv2D(32, kernel\_size=(3, 3),  
 activation='relu',  
 input\_shape=input\_shape)  
conv\_2 = Conv2D(64, (3, 3), activation='relu')  
  
model = Sequential()  
model.add(conv\_1)  
  
model.add(conv\_2)  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(num\_classes, activation='softmax'))

batch\_size = 128  
num\_classes = 10  
epochs = 1

model.fit(x\_train, y\_train,  
 batch\_size=batch\_size,  
 epochs=epochs,  
 verbose=1, validation\_data=(x\_test, y\_test))

两次效果：

Test loss: 0.06023046794650145

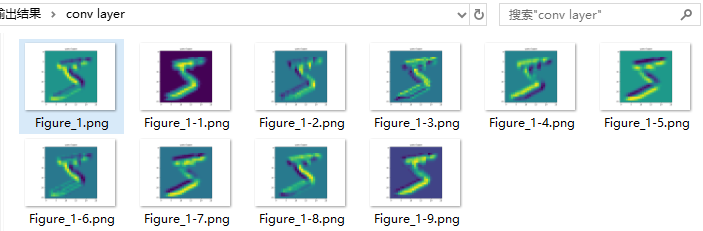
Test accuracy: 0.9816

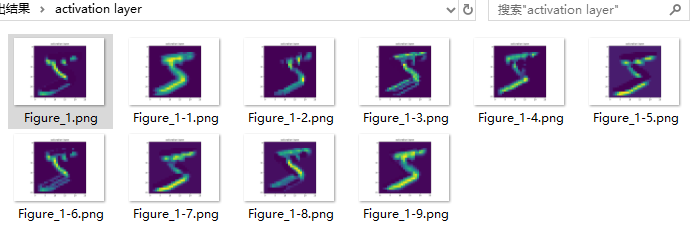
Test loss: 0.06867549542314373

Test accuracy: 0.9775

结论：在同样的epoch下，cnn效果更好，时间更长。Dnn只要10s CNN需要5min

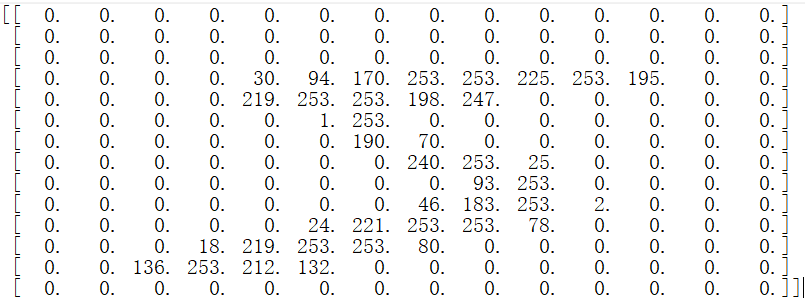
### （1）显示CNN的第一层（卷积层），第二层（激励层）的输出图片(输出其中9张)





**（2）对Mnist中的每个图片做subsampling（研究两种方式，随机采样和抛弃一半），分别利用CNN和DNN在新样本集合作训练，考察subsampling对实验效果的影响（训练误差，测试误差，训练速度），考虑如何减小subsampling的影响。**

1，四倍降采样之后的结果：



CNN：

运行时间：从5min变为为50s。

两次效果：

Test loss: 0.08262749407133088

Test accuracy: 0.9757

Test loss: 0.07995445217452943

Test accuracy: 0.9744

结论：测试误差不大。

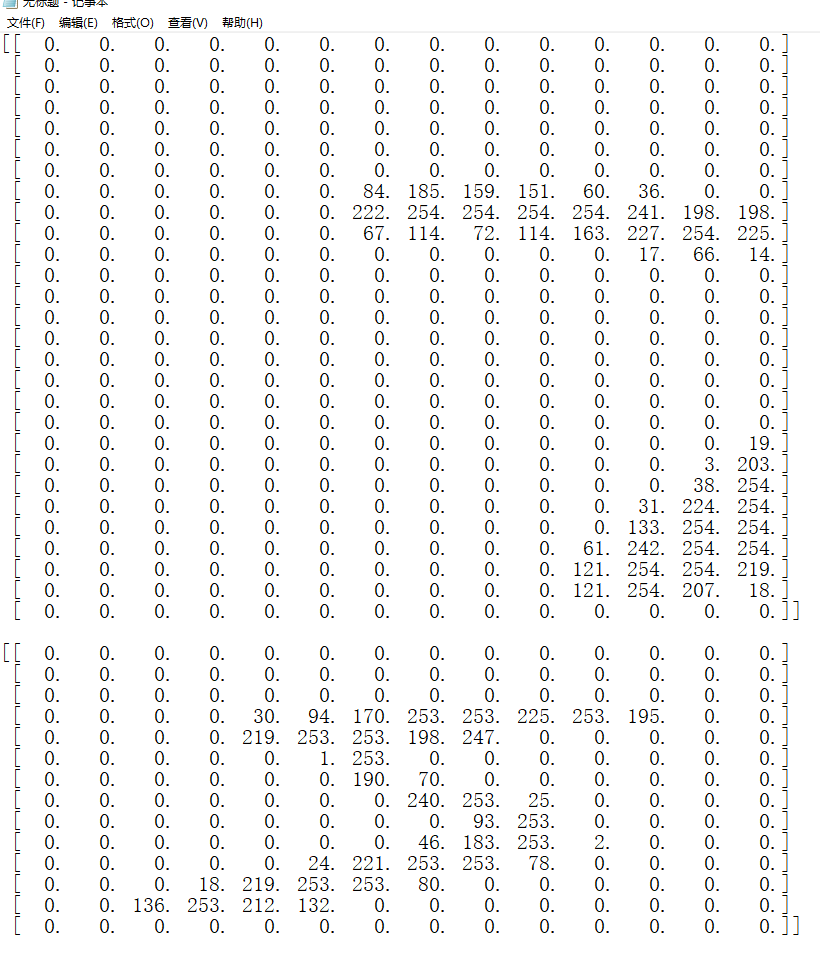
DNN:

训练时间约为5s

test loss: 0.2969126805961132

test accuracy: 0.9169

结论：误差相对仍较大。

2，设置抛弃一半之后的效果：

CNN:

运行时间在2min左右。

两次测试效果：

Test loss: 0.19521159455776216

Test accuracy: 0.9417

Test loss: 0.1949758700400591

Test accuracy: 0.9389

结论：测试误差进一步增大，但是仍保持较高的准确率。

DNN：

训练时间约为5s

test loss: 0.39655784153938295

test accuracy: 0.8881

结论：误差相对CNN更大。

**（3）取原训练集前10%的样本做训练，测试集不变，考察CNN和DNN效果，考虑如何减小小样本集的影响。**

CNN：

两次运行结果：

Test loss: 0.29531892013549804

Test accuracy: 0.9097

Test loss: 0.3230072082400322

Test accuracy: 0.9173

DNN：

两次运行结果：

test loss: 0.4636564645767212

test accuracy: 0.8762

test loss: 0.46389272186756136

test accuracy: 0.8767

通过调整nb\_epoch=10，增加训练次数可优化测试结果：

test loss: 0.27507588347643613

test accuracy: 0.9222

继续增大epoch 将对结果影响不大（修改nb\_epoch=100）：

test loss: 0.62979790314195

test accuracy: 0.9262

**（4）尝试对DNN增加池化层和Dropout层，考察DNN效果变化。**

无pooling层可对DNN池化（维度不一致）。

添加dropout(0.55):

test loss: 0.3060034876406193

test accuracy: 0.9186

添加dropout(0.80):

test loss: 0.5026416866779327

test accuracy: 0.8948

添加dropout(0.99):

test loss: 2.2277009323120116

test accuracy: 0.2207

结论：保持一定比例的dropout不会对结果产生太大影响，速度会有所提升。但是接近1时，将无法识别数据。

**（5）对kernel\_size进行调整（一起调整，或者逐层调整），考察CNN效果变化。思考如何选择kernel\_size**

修改kernel\_size =conv1 (5,5),conv2(3,3):

Test loss: 0.04961018333421089

Test accuracy: 0.9844

结论：影响不大。准确率略有提高。

修改kernel\_size=conv1(2,2),conv2(8,8):

Test loss: 0.04970626147154253

Test accuracy: 0.9824

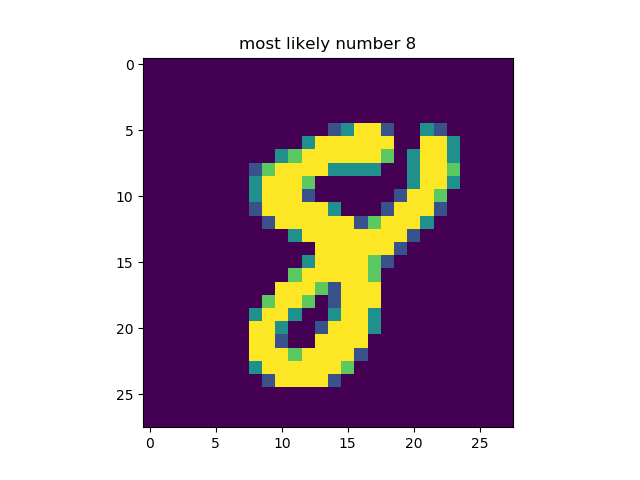
结论：时间消耗为15min，效果无显著提升。可适当提升第一层卷积核，有助于提高识别率。

## 作业任务2：研究如何“欺骗”DNN和CNN

**（1）输出DNN和CNN认为最像“8”的图片**

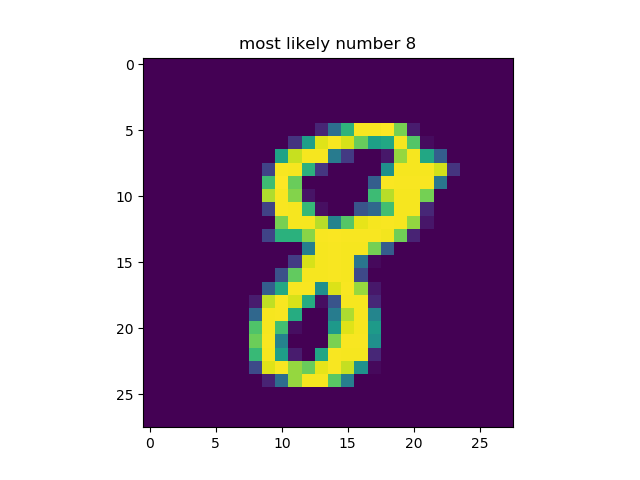
DNN（most likely eight）：

可能性: 0.9992219位置 5003



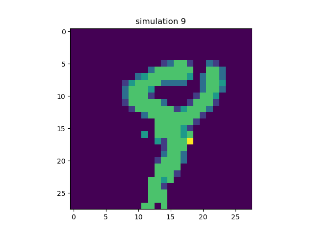
CNN（most likely eight）:

可能性: 0.99999833位置 7114



**（2）在真实“8”图片上加入噪声，让DNN与CNN认为它是“9”**

DNN fake9:

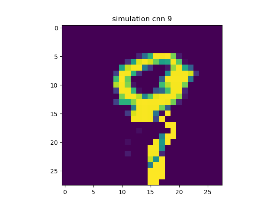


预测值的概率：

[[4.0106301e-07 1.8723180e-05 2.6028445e-05 4.2457259e-04 2.4138275e-04 2.7176596e-03 1.5008907e-06 3.7194110e-02 4.5580325e-01 5.0357234e-01]]

最可能的值： 9

CNN fake9:

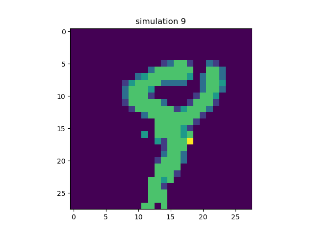


预测值的概率:

[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]

最可能的值： 9

结论：若使用DNN的fake9样例，仍可识别为8



预测值的概率:

[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]

最可能的值： 8

