***Qiuying Li (ql2280)***

***Problem 3***

Python 2.7.13 |Anaconda custom (x86\_64)| (default, Dec 20 2016, 23:05:08)

Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.

? -> Introduction and overview of IPython's features.

%quickref -> Quick reference.

help -> Python's own help system.

object? -> Details about 'object', use 'object??' for extra details.

*# import all the packages;*

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import MaxPooling2D

from keras.utils import np\_utils

from keras import backend as K

*# importing data set*

test = np.loadtxt('/Users/qiuying/Desktop//2017 spring/GR 5241/HW/hw5/zip.test')

train = np.loadtxt('/Users/qiuying/Desktop//2017 spring/GR 5241/HW/hw5/zip.train')

# X variable

training\_x = train[:,1:]

test\_x = test[:,1:]

training\_x = training\_x.reshape(training\_x.shape[0], 1, 16, 16).astype('float32')

#Y variable

training\_y = train[:,0]

test\_y = test[:,0]

test\_x = test\_x.reshape(test\_x.shape[0], 1, 16, 16).astype('float32')

classes\_num = test\_y.shape[1]

*#set random number*

k= ordering('th')

set\_seed = 7

np.random.seed(set\_seed)

#Define the p3\_model as the model we need

def p3\_model():

model = Sequential()

model.add(Conv2D(30, (5, 5), input\_shape=(1, 16, 16), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(15, (3, 3), activation='relu'))

model.add(Flatten())

model.add(Dropout(0.2))

model.add(Dense(50, activation='relu'))

model.add(Dense(128, activation='relu'))

model.add(Dense(classes\_num, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

*# caling the model*

model = p3\_model()

model.fit(training\_x, training\_y, validation\_data=( test\_x, test\_y), epochs=50, batch\_size=200)

Train on 7291 samples, validate on 2007 samples

Epoch 1/50

7291/7291 [==============================] - 6s - loss: 1.8543 - acc: 0.3703 - val\_loss: 1.0001 - val\_acc: 0.7005

Epoch 2/50

7291/7291 [==============================] - 5s - loss: 0.6807 - acc: 0.7738 - val\_loss: 0.4562 - val\_acc: 0.8615

Epoch 3/50

7291/7291 [==============================] - 5s - loss: 0.3902 - acc: 0.8723 - val\_loss: 0.3578 - val\_acc: 0.8934

Epoch 4/50

7291/7291 [==============================] - 5s - loss: 0.2807 - acc: 0.9128 - val\_loss: 0.2869 - val\_acc: 0.9188

Epoch 5/50

7291/7291 [==============================] - 5s - loss: 0.2253 - acc: 0.9317 - val\_loss: 0.2530 - val\_acc: 0.9193

Epoch 6/50

7291/7291 [==============================] - 5s - loss: 0.1892 - acc: 0.9436 - val\_loss: 0.2420 - val\_acc: 0.9307

Epoch 7/50

7291/7291 [==============================] - 5s - loss: 0.1602 - acc: 0.9530 - val\_loss: 0.2214 - val\_acc: 0.9397

Epoch 8/50

7291/7291 [==============================] - 5s - loss: 0.1407 - acc: 0.9534 - val\_loss: 0.2101 - val\_acc: 0.9397

Epoch 9/50

7291/7291 [==============================] - 5s - loss: 0.1269 - acc: 0.9593 - val\_loss: 0.1981 - val\_acc: 0.9422

Epoch 10/50

7291/7291 [==============================] - 5s - loss: 0.1084 - acc: 0.9689 - val\_loss: 0.1836 - val\_acc: 0.9502

Epoch 11/50

7291/7291 [==============================] - 5s - loss: 0.1093 - acc: 0.9649 - val\_loss: 0.1843 - val\_acc: 0.9517

Epoch 12/50

7291/7291 [==============================] - 5s - loss: 0.0860 - acc: 0.9750 - val\_loss: 0.1768 - val\_acc: 0.9552

Epoch 13/50

7291/7291 [==============================] - 5s - loss: 0.0891 - acc: 0.9731 - val\_loss: 0.1614 - val\_acc: 0.9571

Epoch 14/50

7291/7291 [==============================] - 5s - loss: 0.0846 - acc: 0.9727 - val\_loss: 0.1677 - val\_acc: 0.9557

Epoch 15/50

7291/7291 [==============================] - 5s - loss: 0.0785 - acc: 0.9760 - val\_loss: 0.1620 - val\_acc: 0.9567

Epoch 16/50

7291/7291 [==============================] - 5s - loss: 0.0745 - acc: 0.9765 - val\_loss: 0.1840 - val\_acc: 0.9517

Epoch 17/50

7291/7291 [==============================] - 5s - loss: 0.0682 - acc: 0.9800 - val\_loss: 0.1573 - val\_acc: 0.9581

Epoch 18/50

7291/7291 [==============================] - 5s - loss: 0.0637 - acc: 0.9802 - val\_loss: 0.1641 - val\_acc: 0.9611

Epoch 19/50

7291/7291 [==============================] - 5s - loss: 0.0598 - acc: 0.9819 - val\_loss: 0.1683 - val\_acc: 0.9591

Epoch 20/50

7291/7291 [==============================] - 5s - loss: 0.0590 - acc: 0.9807 - val\_loss: 0.1531 - val\_acc: 0.9626

Epoch 21/50

7291/7291 [==============================] - 5s - loss: 0.0545 - acc: 0.9837 - val\_loss: 0.1705 - val\_acc: 0.9567

Epoch 22/50

7291/7291 [==============================] - 5s - loss: 0.0528 - acc: 0.9833 - val\_loss: 0.1637 - val\_acc: 0.9586

Epoch 23/50

7291/7291 [==============================] - 5s - loss: 0.0626 - acc: 0.9813 - val\_loss: 0.1569 - val\_acc: 0.9631

Epoch 24/50

7291/7291 [==============================] - 5s - loss: 0.0508 - acc: 0.9849 - val\_loss: 0.1589 - val\_acc: 0.9591

Epoch 25/50

7291/7291 [==============================] - 5s - loss: 0.0460 - acc: 0.9849 - val\_loss: 0.1611 - val\_acc: 0.9606

Epoch 26/50

7291/7291 [==============================] - 5s - loss: 0.0432 - acc: 0.9877 - val\_loss: 0.1645 - val\_acc: 0.9616

Epoch 27/50

7291/7291 [==============================] - 6s - loss: 0.0487 - acc: 0.9857 - val\_loss: 0.1635 - val\_acc: 0.9606

Epoch 28/50

7291/7291 [==============================] - 5s - loss: 0.0428 - acc: 0.9879 - val\_loss: 0.1605 - val\_acc: 0.9631

Epoch 29/50

7291/7291 [==============================] - 5s - loss: 0.0417 - acc: 0.9866 - val\_loss: 0.1533 - val\_acc: 0.9611

Epoch 30/50

7291/7291 [==============================] - 5s - loss: 0.0446 - acc: 0.9866 - val\_loss: 0.1585 - val\_acc: 0.9616

Epoch 31/50

7291/7291 [==============================] - 5s - loss: 0.0385 - acc: 0.9879 - val\_loss: 0.1657 - val\_acc: 0.9621

Epoch 32/50

7291/7291 [==============================] - 5s - loss: 0.0408 - acc: 0.9867 - val\_loss: 0.1780 - val\_acc: 0.9611

Epoch 33/50

7291/7291 [==============================] - 5s - loss: 0.0388 - acc: 0.9879 - val\_loss: 0.1673 - val\_acc: 0.9616

Epoch 34/50

7291/7291 [==============================] - 6s - loss: 0.0386 - acc: 0.9878 - val\_loss: 0.1762 - val\_acc: 0.9576

Epoch 35/50

7291/7291 [==============================] - 6s - loss: 0.0335 - acc: 0.9904 - val\_loss: 0.1625 - val\_acc: 0.9606

Epoch 36/50

7291/7291 [==============================] - 6s - loss: 0.0371 - acc: 0.9871 - val\_loss: 0.1575 - val\_acc: 0.9656

Epoch 37/50

7291/7291 [==============================] - 6s - loss: 0.0329 - acc: 0.9892 - val\_loss: 0.1677 - val\_acc: 0.9631

Epoch 38/50

7291/7291 [==============================] - 6s - loss: 0.0355 - acc: 0.9883 - val\_loss: 0.1748 - val\_acc: 0.9591

Epoch 39/50

7291/7291 [==============================] - 6s - loss: 0.0340 - acc: 0.9892 - val\_loss: 0.1749 - val\_acc: 0.9646

Epoch 40/50

7291/7291 [==============================] - 5s - loss: 0.0284 - acc: 0.9920 - val\_loss: 0.1672 - val\_acc: 0.9616

Epoch 41/50

7291/7291 [==============================] - 5s - loss: 0.0343 - acc: 0.9888 - val\_loss: 0.1624 - val\_acc: 0.9661

Epoch 42/50

7291/7291 [==============================] - 5s - loss: 0.0352 - acc: 0.9890 - val\_loss: 0.1688 - val\_acc: 0.9616

Epoch 43/50

7291/7291 [==============================] - 5s - loss: 0.0338 - acc: 0.9893 - val\_loss: 0.1765 - val\_acc: 0.9631

Epoch 44/50

7291/7291 [==============================] - 5s - loss: 0.0323 - acc: 0.9894 - val\_loss: 0.1528 - val\_acc: 0.9676

Epoch 45/50

7291/7291 [==============================] - 5s - loss: 0.0251 - acc: 0.9922 - val\_loss: 0.1707 - val\_acc: 0.9666

Epoch 46/50

7291/7291 [==============================] - 5s - loss: 0.0242 - acc: 0.9929 - val\_loss: 0.1885 - val\_acc: 0.9606

Epoch 47/50

7291/7291 [==============================] - 5s - loss: 0.0276 - acc: 0.9916 - val\_loss: 0.1640 - val\_acc: 0.9661

Epoch 48/50

7291/7291 [==============================] - 5s - loss: 0.0273 - acc: 0.9915 - val\_loss: 0.1686 - val\_acc: 0.9661

Epoch 49/50

7291/7291 [==============================] - 5s - loss: 0.0268 - acc: 0.9911 - val\_loss: 0.1642 - val\_acc: 0.9656

Epoch 50/50

7291/7291 [==============================] - 5s - loss: 0.0246 - acc: 0.9927 - val\_loss: 0.1700 - val\_acc: 0.9661

scores = model.evaluate(test\_x, test\_y, verbose=0)

print("Error Rate: %.2f%%" % (100-scores[1]\*100))

Error Rate: 3.39%