import numpy as np

import matplotlib.pyplot as plt

import random

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers.core import Dense, Dropout, Activation

from keras.utils import np\_utils

**Load the MNIST dataset**

In [ ]:

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

print("X\_train shape", X\_train.shape)

print("y\_train shape", y\_train.shape)

print("X\_test shape", X\_test.shape)

print("y\_test shape", y\_test.shape)

X\_train shape (60000, 28, 28)

y\_train shape (60000,)

X\_test shape (10000, 28, 28)

y\_test shape (10000,)

for i in range(9):

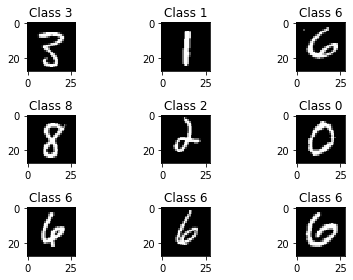
plt.subplot(3,3,i+1)

num = random.randint(0, len(X\_train))

plt.imshow(X\_train[num], cmap='gray', interpolation='none')

plt.title("Class {}".format(y\_train[num]))

plt.tight\_layout()



X\_train = X\_train.reshape(60000, 784)

X\_test = X\_test.reshape(10000, 784)

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

print("Training matrix shape", X\_train.shape)

print("Testing matrix shape", X\_test.shape)

Training matrix shape (60000, 784)

Testing matrix shape (10000, 784)

no\_classes = 10

Y\_train = np\_utils.to\_categorical(y\_train, no\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, no\_classes)

model = Sequential()

model.add(Dense(512, input\_shape=(784,)))

model.add(Activation('relu'))

model.add(Dropout(0.2))

model.add(Dense(512))

model.add(Activation('relu'))

model.add(Dropout(0.2))

model.add(Dense(10))

model.add(Activation('softmax'))

model.summary()

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_7 (Dense) (None, 512) 401920

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_7 (Activation) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_5 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense) (None, 512) 262656

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_8 (Activation) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_6 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_9 (Dense) (None, 10) 5130

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_9 (Activation) (None, 10) 0

=================================================================

Total params: 669,706

Trainable params: 669,706

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

history = model.fit(X\_train, Y\_train,

batch\_size=128, epochs=5,

verbose=1)

Epoch 1/5

60000/60000 [==============================] - 9s 147us/step - loss: 0.2493 - accuracy: 0.9250

Epoch 2/5

60000/60000 [==============================] - 9s 147us/step - loss: 0.1000 - accuracy: 0.9686

Epoch 3/5

60000/60000 [==============================] - 9s 146us/step - loss: 0.0723 - accuracy: 0.9773

Epoch 4/5

60000/60000 [==============================] - 9s 147us/step - loss: 0.0560 - accuracy: 0.9821

Epoch 5/5

60000/60000 [==============================] - 9s 147us/step - loss: 0.0454 - accuracy: 0.9849

score = model.evaluate(X\_test, Y\_test)

print('Test score:', score[0])

print('Test accuracy:', score[1])

10000/10000 [==============================] - 1s 67us/step

Test score: 0.06385218842983595

Test accuracy: 0.9812999963760376

import os

save\_dir = "/"

model\_name = 'keras\_mnist.h5'

model\_path = os.path.join(save\_dir, model\_name)

model.save(model\_path)

print('Saved trained model at %s ' % model\_path)

Saved trained model at /keras\_mnist.h5

fig = plt.figure()

plt.subplot(2,1,1)

plt.plot(history.history['accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)

plt.plot(history.history['loss'])

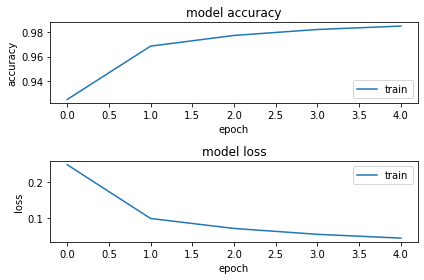
plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper right')

plt.tight\_layout()



predicted\_classes = model.predict\_classes(X\_test)

correct\_indices = np.nonzero(predicted\_classes == y\_test)[0]

incorrect\_indices = np.nonzero(predicted\_classes != y\_test)[0]

In [ ]:

plt.figure()

for i, correct in enumerate(correct\_indices[:9]):

plt.subplot(3,3,i+1)

plt.imshow(X\_test[correct].reshape(28,28), cmap='gray', interpolation='none')

plt.title("Predicted {}, Class {}".format(predicted\_classes[correct], y\_test[correct]))

plt.tight\_layout()

plt.figure()

for i, incorrect in enumerate(incorrect\_indices[:9]):

plt.subplot(3,3,i+1)

plt.imshow(X\_test[incorrect].reshape(28,28), cmap='gray', interpolation='none')

plt.title("Predicted {}, Class {}".format(predicted\_classes[incorrect], y\_test[incorrect]))

plt.tight\_layout()

