import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt # plotting library

%matplotlib inline

from keras.models import Sequential

from keras.layers import Dense , Activation, Dropout

from keras.optimizers import Adam ,RMSprop

from keras.utils import to\_categorical, plot\_model

# import dataset

from keras.datasets import mnist

# load dataset

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

indexes = np.random.randint(0, x\_train.shape[0], size=25)

images = x\_train[indexes]

labels = y\_train[indexes]

# plot the 25 mnist digits

plt.figure(figsize=(5,5))

for i in range(len(indexes)):

plt.subplot(5, 5, i + 1)

image = images[i]

plt.imshow(image, cmap='gray')

plt.axis('off')

plt.show()

plt.savefig("mnist-samples.png")

plt.close('all')

# compute the number of labels

num\_labels = len(np.unique(y\_train))

# convert to one-hot vector

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

# image dimensions (assumed square)

image\_size = x\_train.shape[1]

input\_size = image\_size \* image\_size

input\_size

# resize and normalize

x\_train = np.reshape(x\_train, [-1, input\_size])

x\_train = x\_train.astype('float32') / 255

x\_test = np.reshape(x\_test, [-1, input\_size])

x\_test = x\_test.astype('float32') / 255

# network parameters

batch\_size = 128

hidden\_units = 256

dropout = 0.45

# model is a 3-layer MLP with ReLU and dropout after each layer

model = Sequential()

model.add(Dense(hidden\_units, input\_dim=input\_size))

model.add(Activation('relu'))

model.add(Dropout(dropout))

model.add(Dense(hidden\_units))

model.add(Activation('relu'))

model.add(Dropout(dropout))

model.add(Dense(num\_labels))

model.add(Activation('softmax'))

plot\_model(model, to\_file='mlp-mnist.png', show\_shapes=True)

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

model.fit(x\_train, y\_train, epochs=20, batch\_size=batch\_size)

loss, acc = model.evaluate(x\_test, y\_test, batch\_size=batch\_size)

print("\nTest accuracy: %.1f%%" % (100.0 \* acc))