'''Trains a simple deep NN on the MNIST dataset.

Gets to 98.40% test accuracy after 20 epochs

(there is \*a lot\* of margin for parameter tuning).

2 seconds per epoch on a K520 GPU.

'''

from \_\_future\_\_ import print\_function

import keras

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Dropout

from keras.optimizers import RMSprop

batch\_size = 128

num\_classes = 10

epochs = 5

# the data, split between train and test sets

with np.load('mnist.npz', allow\_pickle=True) as f:

x\_train, y\_train = f['x\_train'], f['y\_train']

x\_test, y\_test = f['x\_test'], f['y\_test']

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(x\_train.shape[0], 'train samples')

print(x\_test.shape[0], 'test samples')

# convert class vectors to binary class matrices one hot

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

model = Sequential()

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

model.add(Dropout(0.2))

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

model.summary()

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

optimizer=RMSprop(),

metrics=['accuracy'])

history = model.fit(x\_train, y\_train,

batch\_size=batch\_size,

epochs=epochs,

verbose=1,(进度条)

validation\_data=(x\_test, y\_test))

score = model.evaluate(x\_test, y\_test, verbose=0)，0显示进度条

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

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