|  |  |
| --- | --- |
| In[1] | import numpy as np  from keras.utils import np\_utils  from keras.datasets import mnist  from matplotlib import pyplot as plt  from keras.models import Sequential # Keras model module  from keras.layers import Dense, Dropout, Activation, Flatten  from keras.layers import Convolution2D, MaxPooling2D |
| Out[1] | Using TensorFlow backend. |

|  |  |
| --- | --- |
| In[2] | import gzip  import pickle  import sys  f = gzip.open('./MNIST\_data/mnist.pkl.gz','rb')  if sys.version\_info < (3,):  data = pickle.load(f)  else:  data = pickle.load(f, encoding='bytes')  f.close()  (x\_train, y\_train), (x\_validation, y\_validation),(x\_test, y\_test) = data |

|  |  |
| --- | --- |
| In[3] | print(x\_train.shape, y\_train.shape, x\_validation.shape, y\_validation.shape,x\_test.shape, y\_test.shape) |
| Out[3] | (50000, 784) (50000,) (10000, 784) (10000,) (10000, 784) (10000,) |

|  |  |
| --- | --- |
| In[4] | x\_train = x\_train.reshape(-1, 1,28, 28)  x\_test = x\_test.reshape(-1, 1,28, 28)  x\_train = x\_train.astype('float32')  x\_test = x\_test.astype('float32')  x\_train /= 255  x\_test /= 25  y\_train = np\_utils.to\_categorical(y\_train, 10)  y\_test = np\_utils.to\_categorical(y\_test,10) |

|  |  |
| --- | --- |
| In[5] | model = Sequential() #init cnn model  #cnn layers  model.add(Convolution2D(32, (5, 5), activation='relu', input\_shape=(1,28,28), data\_format='channels\_first')) #output 282832  model.add(MaxPooling2D(pool\_size=(2,2))) #output 141432  model.add(Convolution2D(64,(5,5),activation='relu'))  model.add(MaxPooling2D(pool\_size=(2,2)))  model.add(Dropout(0.25))  model.add(Flatten())  model.add(Dense(1024, activation='relu'))  model.add(Dropout(0.5))  model.add(Dense(10, activation='softmax'))  model.summary() |
| Out[5] | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  conv2d\_1 (Conv2D) (None, 32, 24, 24) 832  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  max\_pooling2d\_1 (MaxPooling2) (None, 16, 12, 24) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv2d\_2 (Conv2D) (None, 12, 8, 64) 38464  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  max\_pooling2d\_2 (MaxPooling2) (None, 6, 4, 64) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_1 (Dropout) (None, 6, 4, 64) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  flatten\_1 (Flatten) (None, 1536) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_1 (Dense) (None, 1024) 1573888  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_2 (Dropout) (None, 1024) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_2 (Dense) (None, 10) 10250  =================================================================  Total params: 1,623,434  Trainable params: 1,623,434  Non-trainable params: 0 |

|  |  |
| --- | --- |
| In[6] | #model run  model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])  model.fit(x\_train, y\_train, batch\_size=128, epochs=4, verbose=1, validation\_data=(x\_test, y\_test))  score = model.evaluate(x\_test,y\_test, verbose=0) |
| Out[6] | Train on 50000 samples, validate on 10000 samples  Epoch 1/4  50000/50000 [==============================] - 45s 898us/step - loss: 1.0276 - acc: 0.6434 - val\_loss: 1.1273 - val\_acc: 0.8867  Epoch 2/4  50000/50000 [==============================] - 47s 948us/step - loss: 0.2790 - acc: 0.9151 - val\_loss: 0.5540 - val\_acc: 0.9472  Epoch 3/4  50000/50000 [==============================] - 49s 971us/step - loss: 0.1798 - acc: 0.9444 - val\_loss: 0.5669 - val\_acc: 0.9479  Epoch 4/4  50000/50000 [==============================] - 48s 960us/step - loss: 0.1383 - acc: 0.9568 - val\_loss: 0.4318 - val\_acc: 0.9571 |

|  |  |
| --- | --- |
| In[7] | print("accuracy: ",score[1]) |
| Out[7] | accuracy: 0.9571 |