```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
from PIL import Image, ImageOps
import numpy
(X_train, y_train), (X_test, y_test) = mnist.load_data()
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
print(X_train.shape)
print(X_test.shape)
(60000, 28, 28)
(10000, 28, 28)
X_train[0]
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
      0,   0],
[   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
   18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
  253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
  253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
  253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
[   0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
  205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
```

```
[  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
  90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
 190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
 253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
 241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

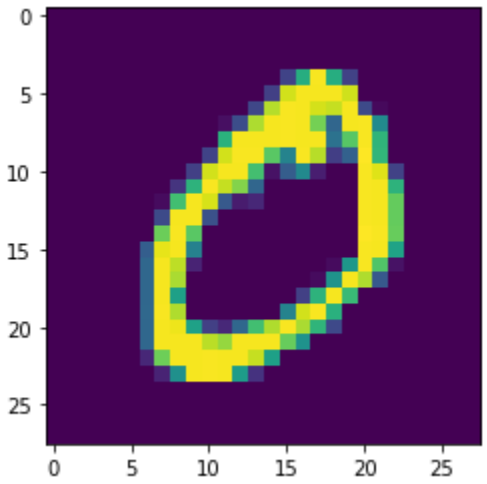
```
      0,   46, 130, 183, 253, 253, 207,      2,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,     39,
    148, 229, 253, 253, 253, 250, 182,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,      0,      0,      0,      0,      0,      0,     24, 114, 221,
    253, 253, 253, 253, 201,    78,      0,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,      0,      0,      0,      0,     23,    66, 213, 253, 253,
    253, 253, 198,    81,      2,      0,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,      0,      0,     18, 171, 219, 253, 253, 253, 253,
    195,    80,      9,      0,      0,      0,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,     55, 172, 226, 253, 253, 253, 253, 244, 133,
     11,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0, 136, 253, 253, 253, 212, 135, 132,    16,      0,
      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
      0,      0],
[     0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
```

```
        0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

```
y_train[0]
```

```
5
```

```
plt.imshow(X_train[1])
```




```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
```

```
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

```
number_of_classes = 10
```

```
Y_train = np_utils.to_categorical(y_train, number_of_classes)
```

```
Y_test = np_utils.to_categorical(y_test, number_of_classes)
```

```
Y_train[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

```
model = Sequential()
```

```
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
```

```
model.add(Conv2D(32, (3, 3), activation="relu"))
```

```
model.add(Flatten())
```

```
model.add(Dense(number_of_classes, activation="softmax"))

model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=["accuracy"])

model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))

Epoch 1/5

1875/1875 [==============================] - 175s 93ms/step - loss: 0.2493 - accuracy: 0.9536 - val_loss: 0.0964 - val_accuracy: 0.9685

Epoch 2/5

1875/1875 [==============================] - 175s 93ms/step - loss: 0.0665 - accuracy: 0.9797 - val_loss: 0.0905 - val_accuracy: 0.9714

Epoch 3/5

1875/1875 [==============================] - 173s 92ms/step - loss: 0.0470 - accuracy: 0.9848 - val_loss: 0.1015 - val_accuracy: 0.9737

Epoch 4/5

1875/1875 [==============================] - 174s 93ms/step - loss: 0.0386 - accuracy: 0.9876 - val_loss: 0.0875 - val_accuracy: 0.9780

Epoch 5/5

1875/1875 [==============================] - 172s 92ms/step - loss: 0.0263 - accuracy: 0.9912 - val_loss: 0.1246 - val_accuracy: 0.9748

metrics = model.evaluate(X_test, Y_test, verbose=0)

print("Metrics (Test Loss & Test Accuracy): ")

print(metrics)

Metrics (Test Loss & Test Accuracy):

[0.1246321052312851, 0.9747999906539917]

prediction = model.predict(X_test[:4])

print(prediction)

1/1 [==============================] - 0s 95ms/step

[[9.69764113e-12 5.35128160e-18 6.73075276e-11 8.16666557e-09
```

```
    6.05515796e-16 1.61785570e-12 8.17041761e-17 1.00000000e+00
    8.81304404e-12 4.62230726e-10]
  [4.07073875e-10 7.69928477e-09 1.00000000e+00 9.22982697e-12
    3.13884536e-17 1.78578504e-20 1.23509398e-08 2.08939884e-17
    3.13647155e-12 7.12971540e-23]
  [4.91924723e-09 9.99989867e-01 7.83750487e-09 2.34650103e-12
    3.13111173e-06 4.78714313e-09 1.26857791e-09 9.65541119e-11
    7.08947118e-06 4.01666284e-10]
  [1.00000000e+00 7.29500773e-15 2.82593549e-11 1.63891306e-16
    1.03110545e-16 8.73841117e-15 1.64485969e-09 6.97776676e-17
    1.71514617e-13 2.39570310e-14]]
print(numpy.argmax(prediction, axis=1))
print(Y_test[:4])
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
model.save("model.h5")
model=load_model("model.h5")
from keras.datasets import mnist
from matplotlib import pyplot
(X_train,y_train),(X_test,y_test)=mnist.load_data()
print('X_train:' +str(X_train.shape))
print('y_train:' +str(y_train.shape))
```

```
print('X_test:' +str(X_test.shape))

print('y_test:' +str(y_test.shape))

from matplotlib import pyplot

for i in range(9):

    pyplot.subplot(330+1+i)

    pyplot.imshow(X_train[i],cmap=pyplot.get_cmap('gray'))

    pyplot.show()
```

```
X_train:(60000, 28, 28)

y_train:(60000,)

X_test:(10000, 28, 28)

y_test:(10000,)
```