```
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()


print(X_train.shape)
print(X_test.shape)
```

```
(60000, 28, 28)
(10000, 28, 28)
```

```
X_train[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,
```

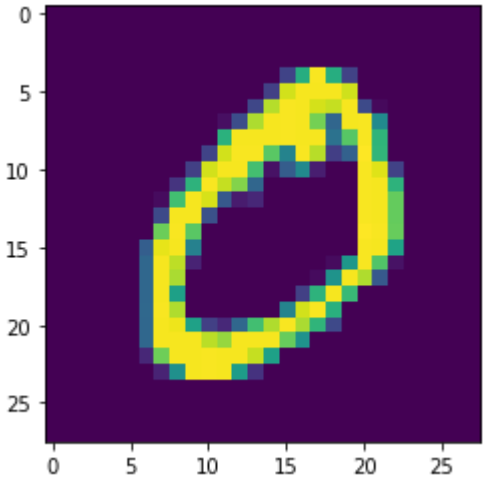
```
 148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
 253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
 253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   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,   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,   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,   0,
   0,   0]], dtype=uint8)
```

```
y_train[0]
```

```
5
```

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

```
<matplotlib.image.AxesImage at 0x7fdad3e8ca90>
```




```
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 [==============================] - 201s 107ms/step - loss: 0.2706 - a
Epoch 2/5
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0754 - a
Epoch 3/5
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0524 - a
Epoch 4/5
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0384 - a
Epoch 5/5
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0284 - a
<keras.callbacks.History at 0x7fdad58d9550>
```

```
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.11876353621482849, 0.9739999771118164]
```

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

```
1/1 [==============================] - 0s 92ms/step
[[3.96809002e-10 4.67274070e-19 4.86880491e-09 1.87814621e-07
  7.87748665e-18 4.43033680e-15 9.73031877e-20 9.99999762e-01
  4.88716538e-15 5.05039621e-11]
 [6.82645260e-11 2.37826859e-12 1.00000000e+00 1.60329042e-10
  1.24526496e-16 8.01968406e-20 9.42386967e-12 7.04118420e-19
  3.98000209e-12 3.71121166e-17]
 [3.89413231e-08 9.99972105e-01 7.21178282e-08 1.16173848e-09
  1.44862133e-05 2.52747412e-07 7.77183004e-07 1.11785807e-08
  1.22595975e-05 3.13419538e-12]
 [9.99999285e-01 2.67883951e-18 5.09310325e-07 3.38718946e-15
```

```
        3.75197226e-12 2.78726459e-10 5.06628917e-09 1.52035849e-12
        1.80088009e-07 4.17252760e-10]]
```

```
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")
```