```python
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
```

```python
(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
```

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

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

```python
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,   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,
```

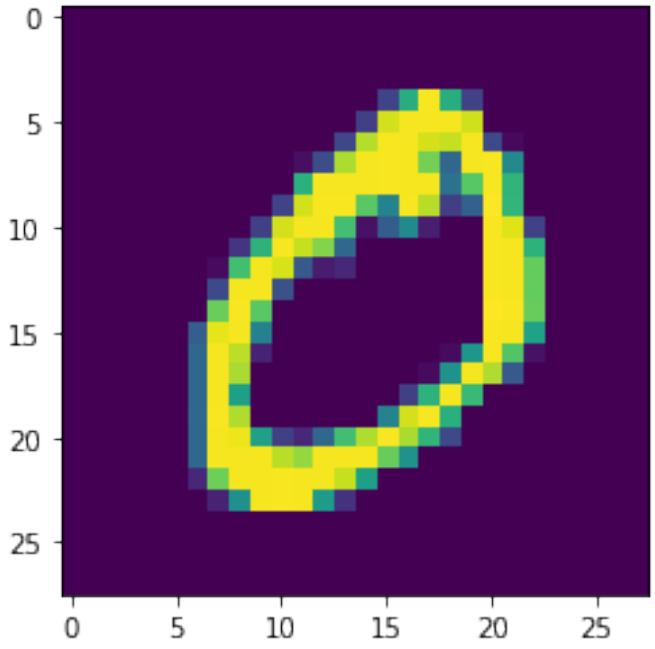
```
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
           0,   0],
       [   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
           0,   0],
       [   0,   0,   0,   0,   0,   0,   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 0x7fc6ac439910>
```




```
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 [==============================] - 198s 105ms/step - loss:
0.2459 - accuracy: 0.9522 - val_loss: 0.1057 - val_accuracy: 0.9684
Epoch 2/5
1875/1875 [==============================] - 198s 106ms/step - loss:
0.0703 - accuracy: 0.9791 - val_loss: 0.0762 - val_accuracy: 0.9766
Epoch 3/5
1875/1875 [==============================] - 195s 104ms/step - loss:
0.0501 - accuracy: 0.9844 - val_loss: 0.1054 - val_accuracy: 0.9676
Epoch 4/5
1875/1875 [==============================] - 194s 104ms/step - loss:
0.0379 - accuracy: 0.9877 - val_loss: 0.0942 - val_accuracy: 0.9765
Epoch 5/5
1875/1875 [==============================] - 195s 104ms/step - loss:
0.0284 - accuracy: 0.9905 - val_loss: 0.0903 - val_accuracy: 0.9783

<keras.callbacks.History at 0x7fc6a4d6e910>

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

Metrics (Test Loss & Test Accuracy):
[0.09031817317008972, 0.9782999753952026]

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

1/1 [==============================] - 0s 131ms/step
[[2.22217793e-13 6.39153619e-19 1.88266937e-11 4.34683817e-10
```

```
  4.45043487e-19 2.45938845e-17 3.99081647e-21 1.00000000e+00
  8.00842836e-10 4.09856307e-12]
 [1.10607878e-07 1.48146322e-08 9.99998450e-01 2.25929206e-13
  9.84383690e-13 1.68875632e-16 1.42071372e-06 5.39981651e-16
  9.75629910e-10 1.89072023e-22]
 [1.40427892e-08 9.99861717e-01 6.96560573e-06 9.03956109e-13
  1.27997419e-05 2.59406818e-07 6.22464427e-07 1.15367424e-04
  2.20492007e-06 8.93586805e-09]
 [1.00000000e+00 3.40269377e-15 8.21864590e-11 5.48245410e-16
  4.15226013e-11 2.52093593e-13 1.82735711e-08 1.76641930e-11
  1.56697488e-10 1.48902224e-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")
```