```
import numpy
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
```

```
(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 [==============================] - 1s 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,   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,
```

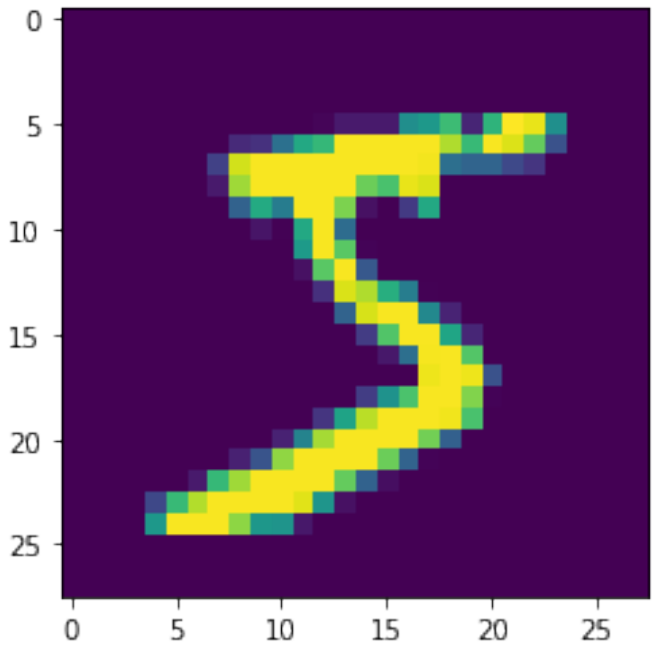
```
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0,   0,   0,   0,   0,   0,   0,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

```
y_train[0]
```

```
5
```

```
plt.imshow(X_train[0])
```

```
<matplotlib.image.AxesImage at 0x7f9d32125e90>
```




```
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 [==============================] - 172s 91ms/step - loss:
0.2263 - accuracy: 0.9507 - val_loss: 0.0857 - val_accuracy: 0.9730
Epoch 2/5
1875/1875 [==============================] - 169s 90ms/step - loss:
0.0688 - accuracy: 0.9789 - val_loss: 0.0787 - val_accuracy: 0.9753
Epoch 3/5
1875/1875 [==============================] - 169s 90ms/step - loss:
0.0480 - accuracy: 0.9842 - val_loss: 0.0780 - val_accuracy: 0.9778
Epoch 4/5
1875/1875 [==============================] - 169s 90ms/step - loss:
0.0390 - accuracy: 0.9876 - val_loss: 0.0836 - val_accuracy: 0.9777
Epoch 5/5
1875/1875 [==============================] - 170s 91ms/step - loss:
0.0275 - accuracy: 0.9910 - val_loss: 0.1014 - val_accuracy: 0.9780

<keras.callbacks.History at 0x7f9d2d83e5d0>

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

Metrics (Test Loss & Test Accuracy):
[0.10139955580234528, 0.9779999852180481]

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

1/1 [==============================] - 0s 95ms/step
[[4.84883661e-12 5.46598212e-22 6.38604729e-13 1.86603987e-13
  8.73846900e-17 3.01616625e-19 5.99638645e-22 1.00000000e+00
  9.21016085e-13 3.41176774e-14]
 [5.71149217e-09 6.21804830e-10 9.99999881e-01 1.10658705e-14
  5.36142962e-13 2.28247958e-18 1.39807142e-07 2.48846980e-20
  3.51173188e-11 2.77617504e-19]
 [7.46725082e-09 9.99766290e-01 1.33248068e-06 1.38112300e-13
  1.89813683e-04 4.70011770e-07 3.24271432e-09 1.22552152e-07
  4.19943572e-05 6.25929083e-12]
 [1.00000000e+00 4.12239659e-16 9.50166834e-10 9.96371717e-15
```

```
  7.47916225e-12 6.28938523e-15 1.59912936e-08 5.95622130e-13
  9.92760778e-13 6.20633892e-11]]
```

```
print(numpy.argmax(prediction, axis=1))
print(Y_test[:6])
```

```
[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.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
```