

```
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-data
sets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
```

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

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

```
X_train[0]
```

```
Out[4]: 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,  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, 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,  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,    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,    0,
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,    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, 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, 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]], dtype=uint8)

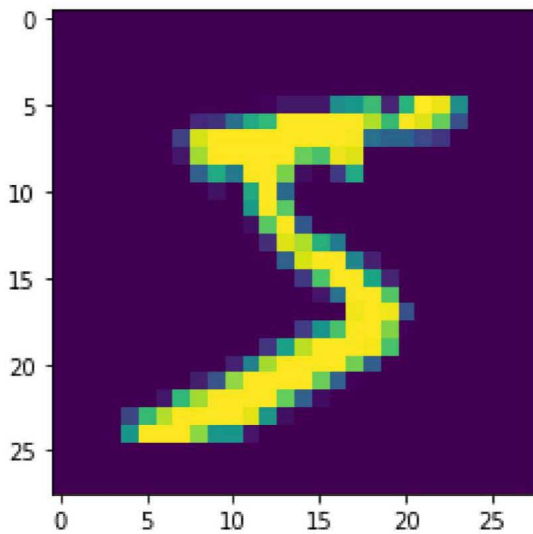
```

In [5]: `y_train[0]`

Out[5]: 5

```
In [6]: plt.imshow(X_train[0])
```

Out[6]:



```
In [7]: X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

```
In [8]: 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)
```

```
In [9]: Y_train[0]
```

Out[9]: array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)

```
In [10]: 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"))
```

```
In [11]: model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=["a
```

```
In [12]: model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test
```

```
Epoch 1/5
1875/1875 [=====] - 192s 102ms/step - loss: 0.2245 -
accuracy: 0.9518 - val_loss: 0.1058 - val_accuracy: 0.9701
Epoch 2/5
1875/1875 [=====] - 197s 105ms/step - loss: 0.0685 -
accuracy: 0.9788 - val_loss: 0.0962 - val_accuracy: 0.9752
Epoch 3/5
1875/1875 [=====] - 190s 101ms/step - loss: 0.0468 -
accuracy: 0.9854 - val_loss: 0.0900 - val_accuracy: 0.9749
Epoch 4/5
```

```

1875/1875 [=====] - 190s 102ms/step - loss: 0.0351 -
accuracy: 0.9891 - val_loss: 0.0993 - val_accuracy: 0.9748
Epoch 5/5
1875/1875 [=====] - 191s 102ms/step - loss: 0.0270 -
accuracy: 0.9917 - val_loss: 0.1005 - val_accuracy: 0.9764

```

Out[12]:

```

In [13]: metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)

```

```

Metrics (Test Loss & Test Accuracy):
[0.10052110999822617, 0.9764000177383423]

```

```

In [14]: prediction = model.predict(X_test[:4])
print(prediction)

```

```

1/1 [=====] - 0s 92ms/step
[[1.5678695e-09 1.6640128e-14 2.0494097e-12 1.5698962e-08 5.4015579e-15
 3.6338055e-13 2.2240399e-20 1.0000000e+00 2.9577885e-08 1.9005494e-08]
 [5.8188578e-09 1.2512093e-10 9.9999821e-01 7.4831279e-09 1.0770124e-10
 2.9252167e-18 1.6483800e-06 1.5410843e-14 1.2811967e-07 3.3103555e-12]
 [1.2689595e-09 9.9028254e-01 3.9091717e-08 1.3732340e-10 9.6216686e-03
 2.9094124e-07 1.9340013e-10 4.5208512e-07 9.5003670e-05 2.4108826e-10]
 [1.0000000e+00 7.3556976e-16 3.5439882e-12 4.7910155e-14 3.2022885e-12
 1.5000925e-12 1.5939531e-11 4.1566353e-14 7.7353792e-12 1.2456662e-09]]

```

```

In [17]: 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.]]

```

```

In [18]: model.save("model.h5")

```

```

In [19]: model=load_model("model.h5")

```

```

In [34]: 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,)
```

