

```
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/mnist11490434/11490434> [=====] - 0s 0us/step

```
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, 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,  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,  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,  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,  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,
  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,  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,  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,
  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, 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]], dtype=uint8)

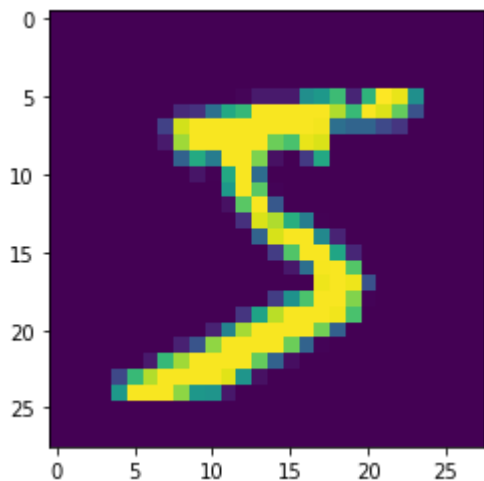
```

```
y_train[0]
```

```
5
```

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

```
<matplotlib.image.AxesImage at 0x7f214610e850>
```



```
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 [=====] - 170s 90ms/step - loss: 0.2343 - accuracy
Epoch 2/5
1875/1875 [=====] - 168s 90ms/step - loss: 0.0709 - accuracy
Epoch 3/5
1875/1875 [=====] - 167s 89ms/step - loss: 0.0473 - accuracy
Epoch 4/5
1875/1875 [=====] - 167s 89ms/step - loss: 0.0361 - accuracy
Epoch 5/5
1875/1875 [=====] - 167s 89ms/step - loss: 0.0295 - accuracy
<keras.callbacks.History at 0x7f214196ee90>
```



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

```
Metrics (Test Loss & Test Accuracy):
[0.110375314950943, 0.9736999869346619]
```

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

```
1/1 [=====] - 0s 85ms/step
[[1.0014383e-08 3.8589627e-15 7.5520370e-06 4.6376817e-07 1.9088618e-14
 1.4835914e-15 1.1017242e-17 9.9999177e-01 6.8098577e-10 2.8772141e-07]
 [1.1860008e-11 3.2682312e-08 9.9999928e-01 3.0327659e-12 1.4758742e-16
 7.2239602e-18 6.8433309e-07 2.4694202e-15 3.5512822e-11 9.6993520e-16]
 [2.5998781e-07 9.9969757e-01 6.9102018e-05 1.7349749e-09 9.0480444e-06
 2.3576348e-08 7.5362621e-08 8.1930702e-09 2.2388414e-04 1.8452234e-09]
 [1.0000000e+00 3.9643531e-14 2.3053618e-11 3.3599832e-13 2.5623741e-13
 2.9599706e-13 2.4084601e-09 2.9738139e-12 4.5129538e-12 1.3925914e-13]]
```

```
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.]]
```

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 8:16 PM

● ✕