```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import matplotlib.pyplot as plt
#from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Conv2D, Dense, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import mnist
```

```
(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
```

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

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

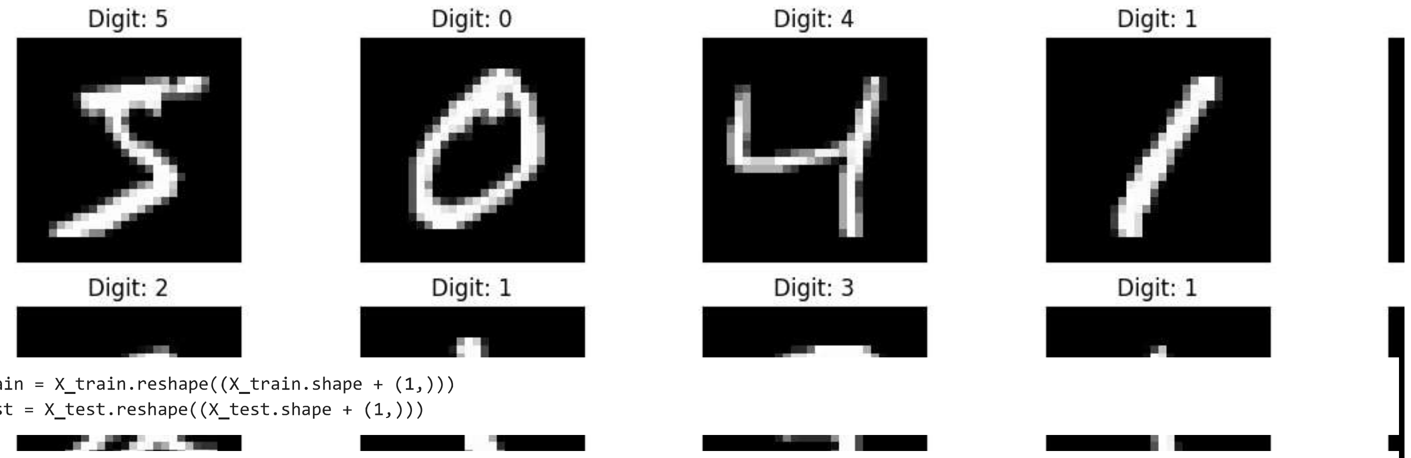
```
X_train[0].min(), X_train[0].max()
```

```
(0, 255)
```

```
X_train = (X_train - 0.0) / (255.0 - 0.0)
X_test = (X_test - 0.0) / (255.0 - 0.0)
X_train[0].min(), X_train[0].max()
```

```
(0.0, 1.0)
```

```
def plot_digit(image, digit, plt, i):
    plt.subplot(4, 5, i + 1)
    plt.imshow(image, cmap=plt.get_cmap('gray'))
    plt.title(f"Digit: {digit}")
    plt.xticks([])
    plt.yticks([])
plt.figure(figsize=(16, 10))
for i in range(20):
    plot_digit(X_train[i], y_train[i], plt, i)
plt.show()
```



```
X_train = X_train.reshape((X_train.shape + (1,)))
X_test = X_test.reshape((X_test.shape + (1,)))
```

```
y_train[0:20]
```

```
     array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4, 3, 5, 3, 6, 1, 7, 2, 8, 6, 9],
           dtype=uint8)
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation="relu", input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(100, activation="relu"),
    Dense(10, activation="softmax")
])
```

```
optimizer = SGD(learning_rate=0.01, momentum=0.9)
model.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
model.summary()
```

```
     Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      conv2d (Conv2D)             (None, 26, 26, 32)        320

      max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
      D)

      flatten (Flatten)           (None, 5408)              0

      dense (Dense)               (None, 100)               540900

      dense_1 (Dense)             (None, 10)                1010

     =================================================================
     Total params: 542230 (2.07 MB)
     Trainable params: 542230 (2.07 MB)
     Non-trainable params: 0 (0.00 Byte)
     _________________________________________________________________
```

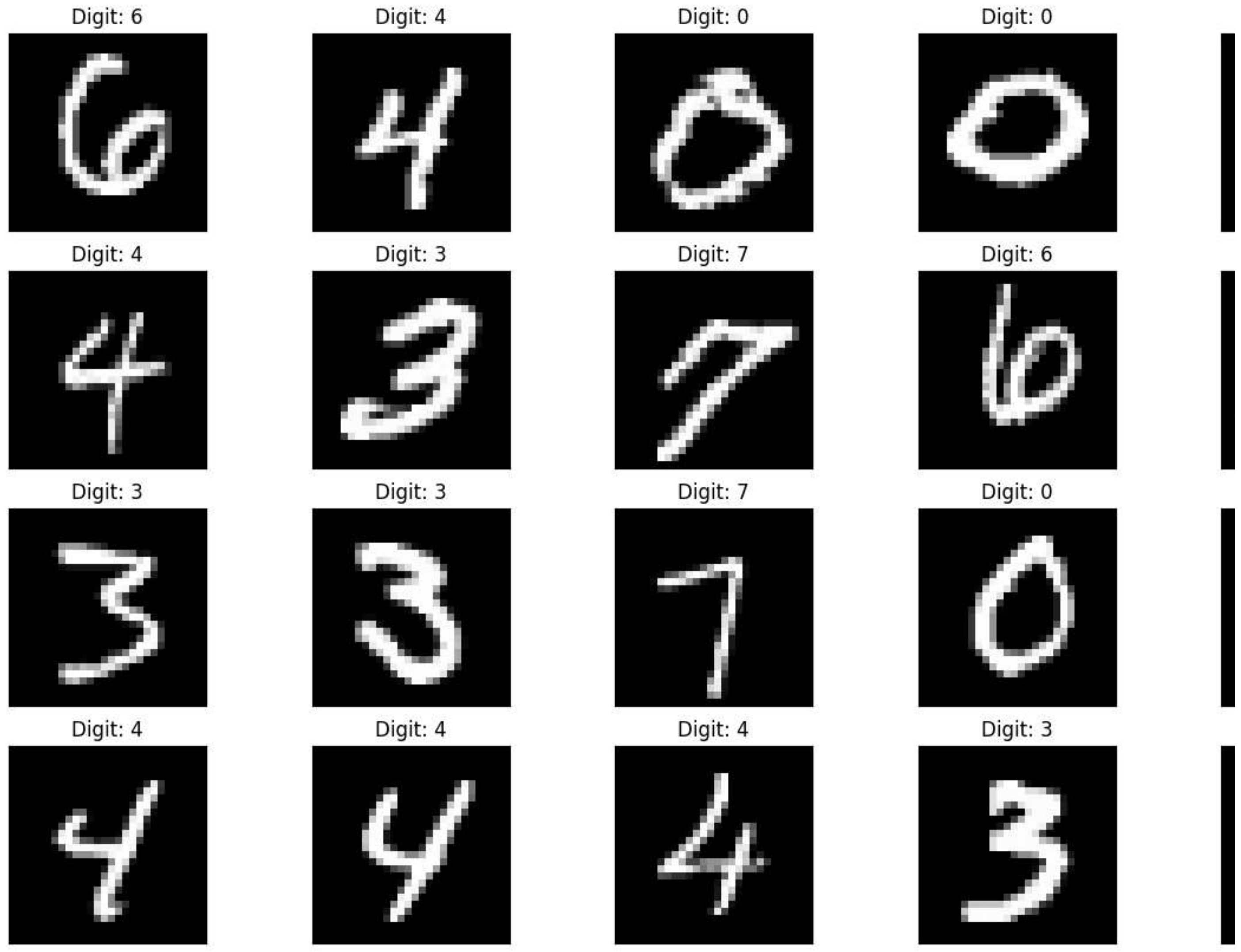
```
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
     Epoch 1/10
     1875/1875 [==============================] - 34s 18ms/step - loss: 0.2356 - accuracy: 0.9285
     Epoch 2/10
     1875/1875 [==============================] - 31s 16ms/step - loss: 0.0765 - accuracy: 0.9766
     Epoch 3/10
     1875/1875 [==============================] - 32s 17ms/step - loss: 0.0488 - accuracy: 0.9849
     Epoch 4/10
     1875/1875 [==============================] - 30s 16ms/step - loss: 0.0354 - accuracy: 0.9891
     Epoch 5/10
     1875/1875 [==============================] - 31s 16ms/step - loss: 0.0260 - accuracy: 0.9918
     Epoch 6/10
     1875/1875 [==============================] - 29s 16ms/step - loss: 0.0195 - accuracy: 0.9937
     Epoch 7/10
     1875/1875 [==============================] - 29s 16ms/step - loss: 0.0141 - accuracy: 0.9961
     Epoch 8/10
```

```
1875/1875 [==============================] - 29s 16ms/step - loss: 0.0121 - accuracy: 0.9963
Epoch 9/10
1875/1875 [==============================] - 33s 17ms/step - loss: 0.0085 - accuracy: 0.9975
Epoch 10/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0064 - accuracy: 0.9983
<keras.src.callbacks.History at 0x7c5485984970>
```

```
plt.figure(figsize=(16, 10))
for i in range(20):
    image = random.choice(X_test).squeeze()
    digit = np.argmax(model.predict(image.reshape((1, 28, 28, 1)))[0], axis=-1)
    plot_digit(image, digit, plt, i)
plt.show()
```

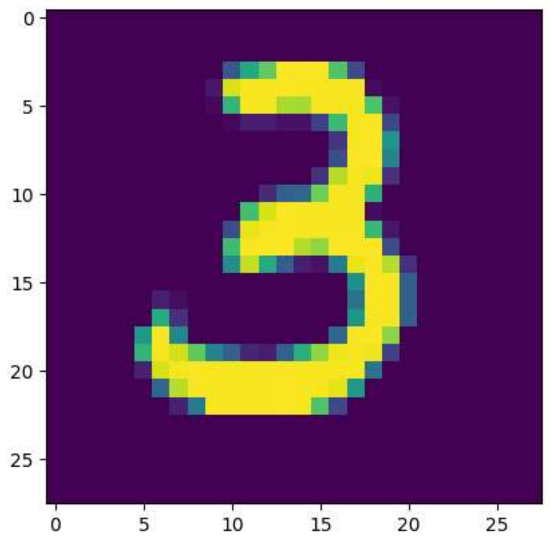
```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
```



```
predictions = np.argmax(model.predict(X_test), axis=-1)
accuracy_score(y_test, predictions)
```

```
313/313 [==============================] - 4s 11ms/step
0.988
```

```
n=random.randint(0,9999)
plt.imshow(X_test[n])
plt.show()
```



```
predicted_value=model.predict(X_test)
print("Handwritten number in the image is= %d" %np.argmax(predicted_value[n]))
```

```
313/313 [==============================] - 2s 5ms/step
Handwritten number in the image is= 3
```

```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0]) #Test loss: 0.0296396646054
print('Test accuracy:', score[1])
```

```
Test loss: 0.04349197819828987
Test accuracy: 0.9879999756813049
```