In [1]:
```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
from matplotlib import*
import matplotlib.pyplot as plt
```

In [2]:
```
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
from tensorflow.keras import Model
from tensorflow.keras.models import Model
```

In [3]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [4]:
```
print(x_train.shape)
```

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

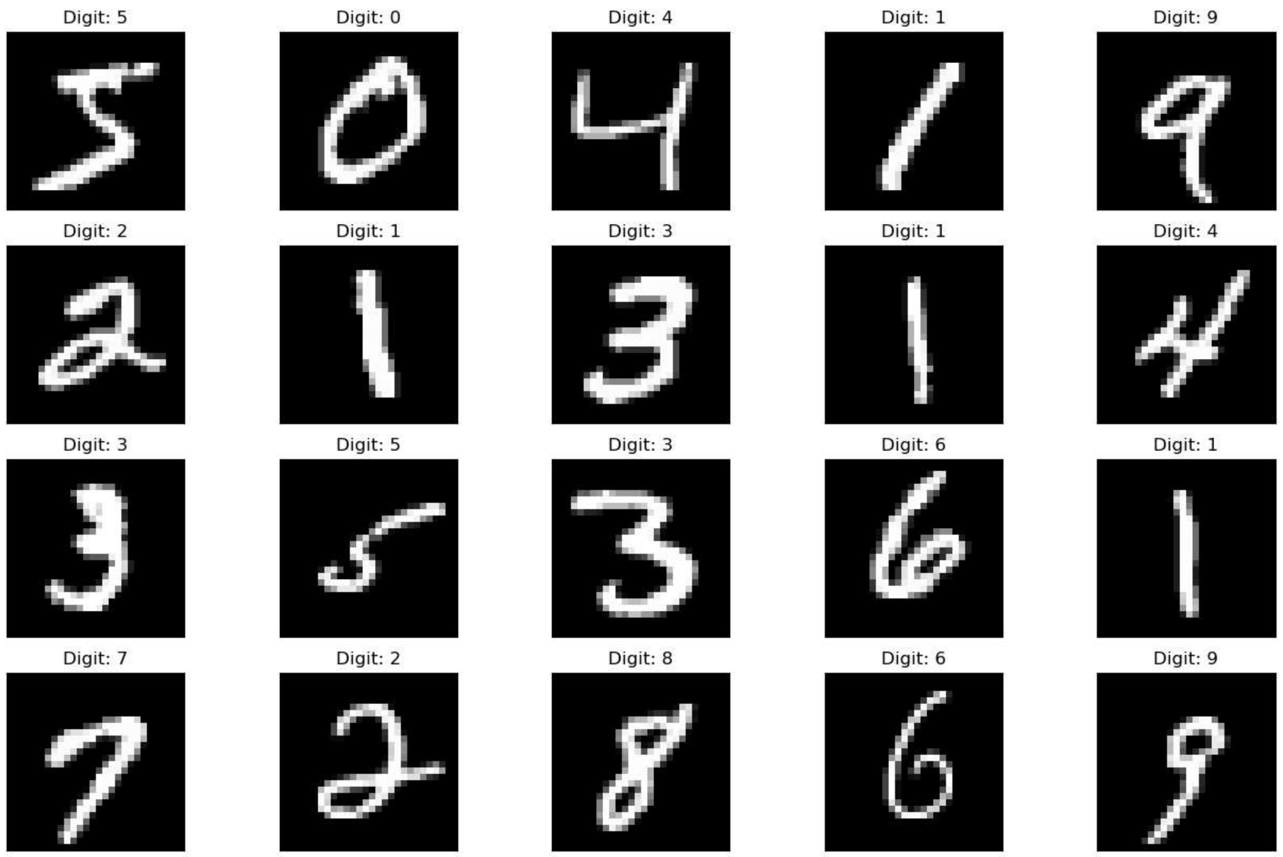
In [5]:
```
x_train[0].min(), x_train[0].max()
```

Out[5]:
```
(0, 255)
```

In [6]:
```
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)
```

Out[6]:
```
(0.0, 1.0)
```

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



In [9]:
```
x_train = x_train.reshape((x_train.shape + (1,)))
x_test = x_test.reshape((x_test.shape + (1,)))
```

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

Out[10]:
```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4, 3, 5, 3, 6, 1, 7, 2, 8, 6, 9],
      dtype=uint8)
```

In [11]:
```
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")
])
```

In [12]:
```
optimizer = SGD(learning_rate=0.01, momentum=0.9)
model.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

In [13]:
```
model.summary()
```

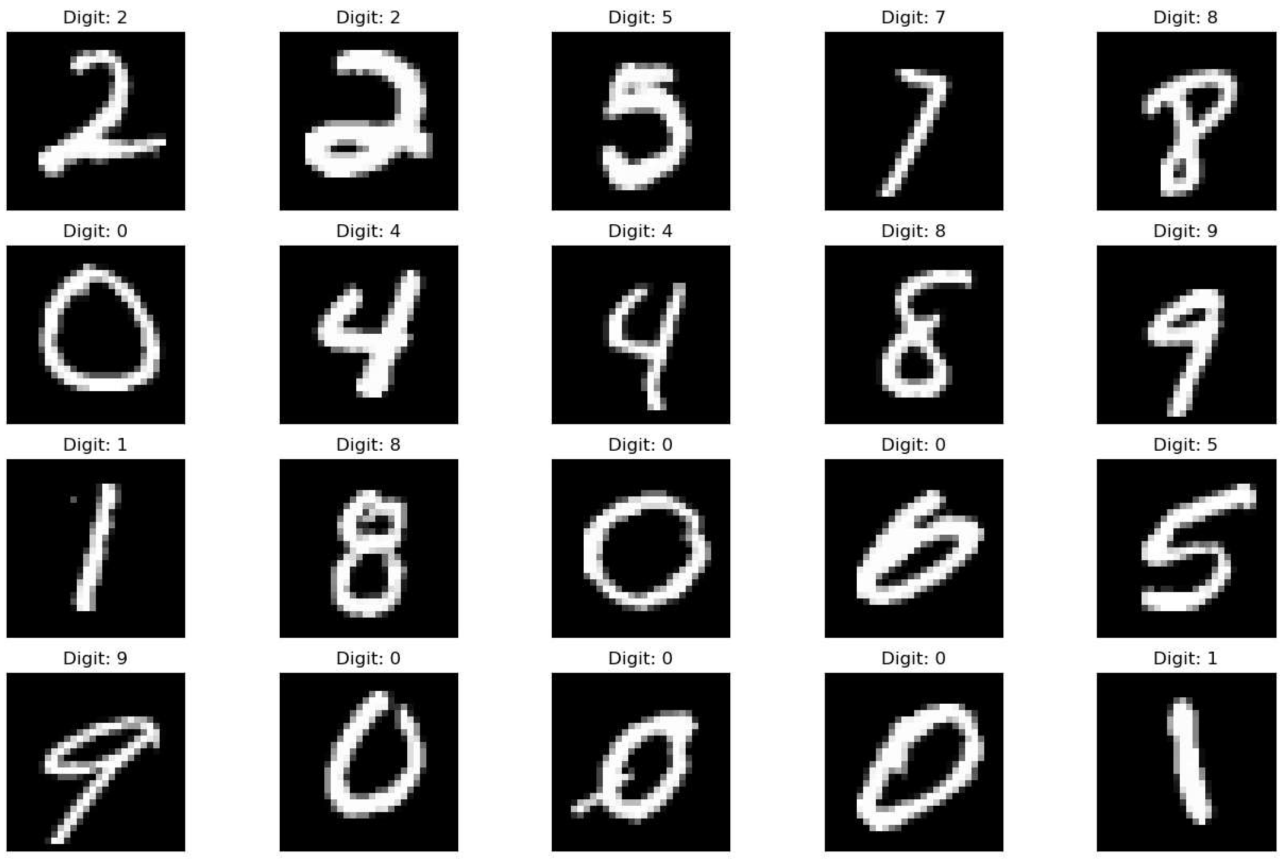
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 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)
_________________________________________________________________
```

In [14]:
```
history=model.fit(x_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
1875/1875 [==============================] - 44s 23ms/step - loss: 0.2354 - accuracy: 0.9299
Epoch 2/10
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0789 - accuracy: 0.9760
Epoch 3/10
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0507 - accuracy: 0.9848
Epoch 4/10
1875/1875 [==============================] - 42s 23ms/step - loss: 0.0358 - accuracy: 0.9893
Epoch 5/10
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0264 - accuracy: 0.9921
Epoch 6/10
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0196 - accuracy: 0.9937
Epoch 7/10
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0141 - accuracy: 0.9959
Epoch 8/10
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0101 - accuracy: 0.9973
Epoch 9/10
1875/1875 [==============================] - 42s 23ms/step - loss: 0.0080 - accuracy: 0.9978
Epoch 10/10
1875/1875 [==============================] - 45s 24ms/step - loss: 0.0056 - accuracy: 0.9988
```

In [15]:
```
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 312ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 84ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 48ms/step
```



In [16]:
```
predictions = np.argmax(model.predict(x_test), axis=-1)
accuracy_score(y_test, predictions)
```

```
313/313 [==============================] - 3s 9ms/step
```

Out[16]:
```
0.9878
```

In [17]:
```
score=model.evaluate(x_test,y_test,verbose=0)
```

In [18]:
```
print('testloss:',score[0])
print('Test accuracy:',score[1])
```

```
testloss: 0.044720057398080826
Test accuracy: 0.9878000020980835
```

In [ ]: