```python
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import matplotlib.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
```

```python
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```python
print(X_train.shape)
```
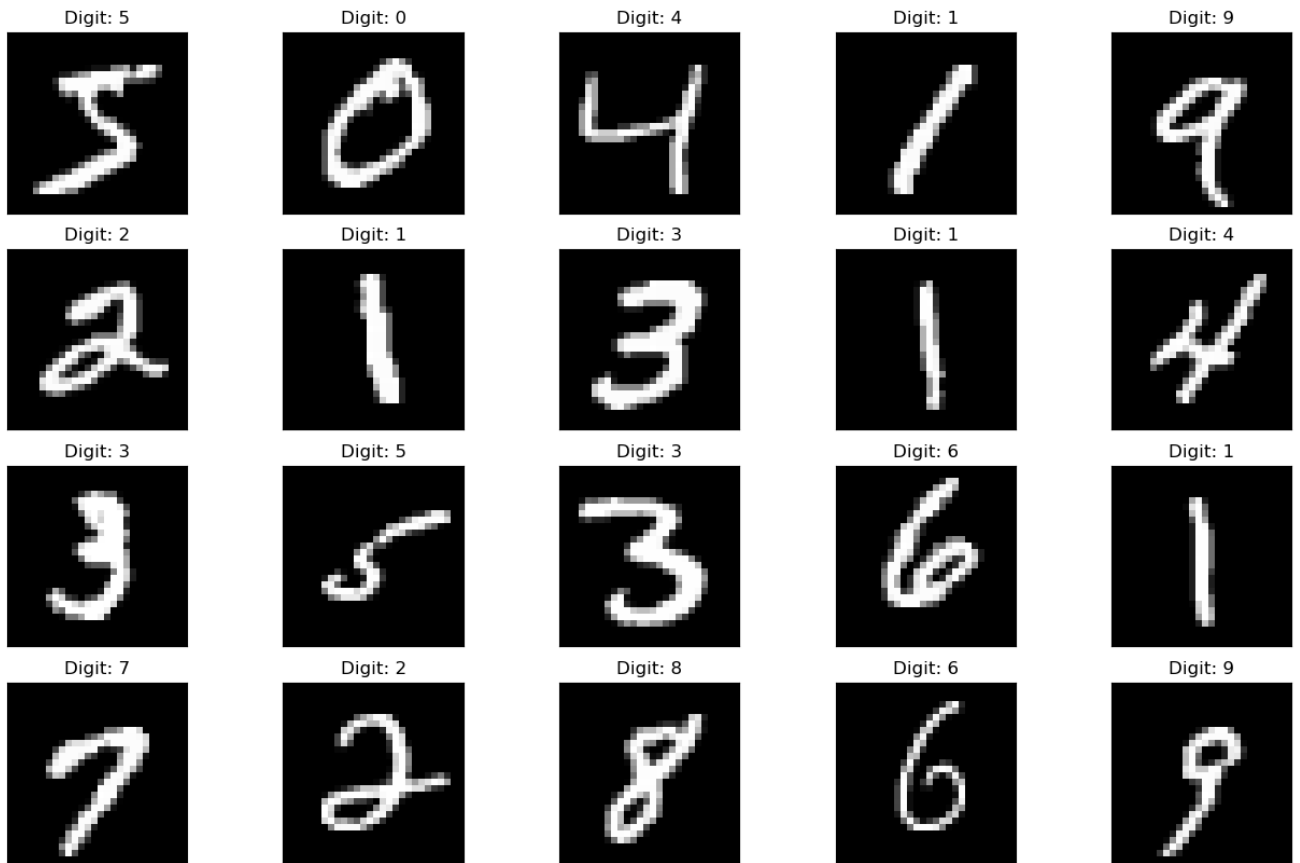
```
(60000, 28, 28)
```

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

```
(0, 255)
```

```python
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)
```

```python
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 [7]:
```python
X_train = X_train.reshape((X_train.shape + (1,)))
X_test = X_test.reshape((X_test.shape + (1,)))
```

In [8]:
```python
y_train[0:20]
```

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

In [9]:
```python
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 [10]:
```python
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 (MaxPooling2D)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0

```
dense (Dense)                    (None, 100)              540900
_____
dense_1 (Dense)                  (None, 10)               1010
=================================================================
Total params: 542,230
Trainable params: 542,230
Non-trainable params: 0
_____
```

In [11]:
```python
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.2314 - accuracy: 0.9305
Epoch 2/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.0747 - accuracy: 0.9768
Epoch 3/10
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0480 - accuracy: 0.9855
Epoch 4/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.0345 - accuracy: 0.9891
Epoch 5/10
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0248 - accuracy: 0.9922
Epoch 6/10
1875/1875 [==============================] - 12s 7ms/step - loss: 0.0190 - accuracy: 0.9941
Epoch 7/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.0139 - accuracy: 0.9960
Epoch 8/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.0105 - accuracy: 0.9971
Epoch 9/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.0069 - accuracy: 0.9986
Epoch 10/10
1875/1875 [==============================] - 13s 7ms/step - loss: 0.0055 - accuracy: 0.9989
```

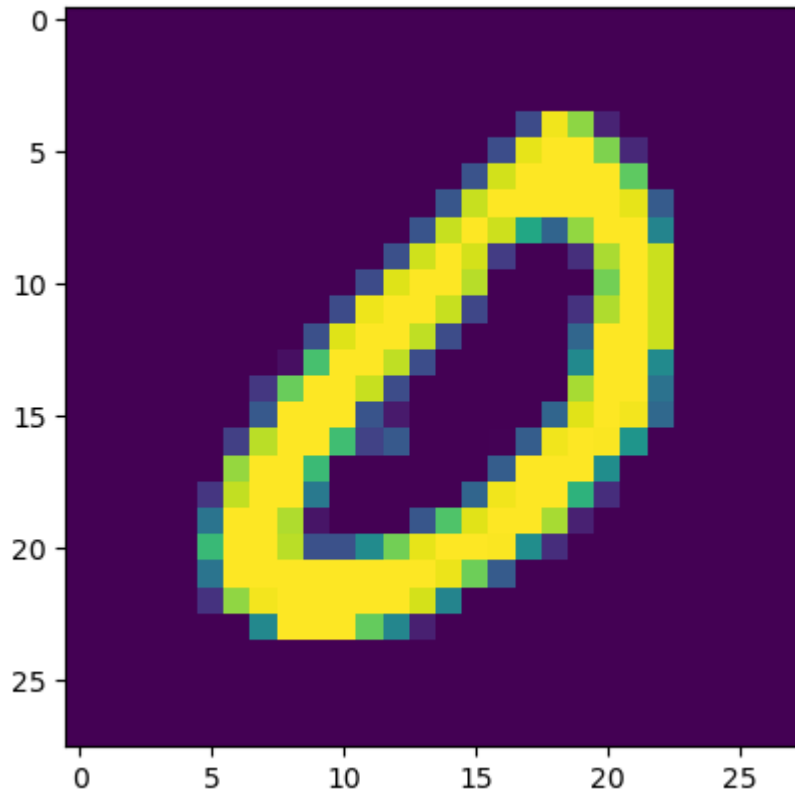Out [11]: <tensorflow.python.keras.callbacks.History at 0x29e06f3d948>

In [12]:
```python
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
    plot_digit(image, digit, plt, i)
plt.show()
```
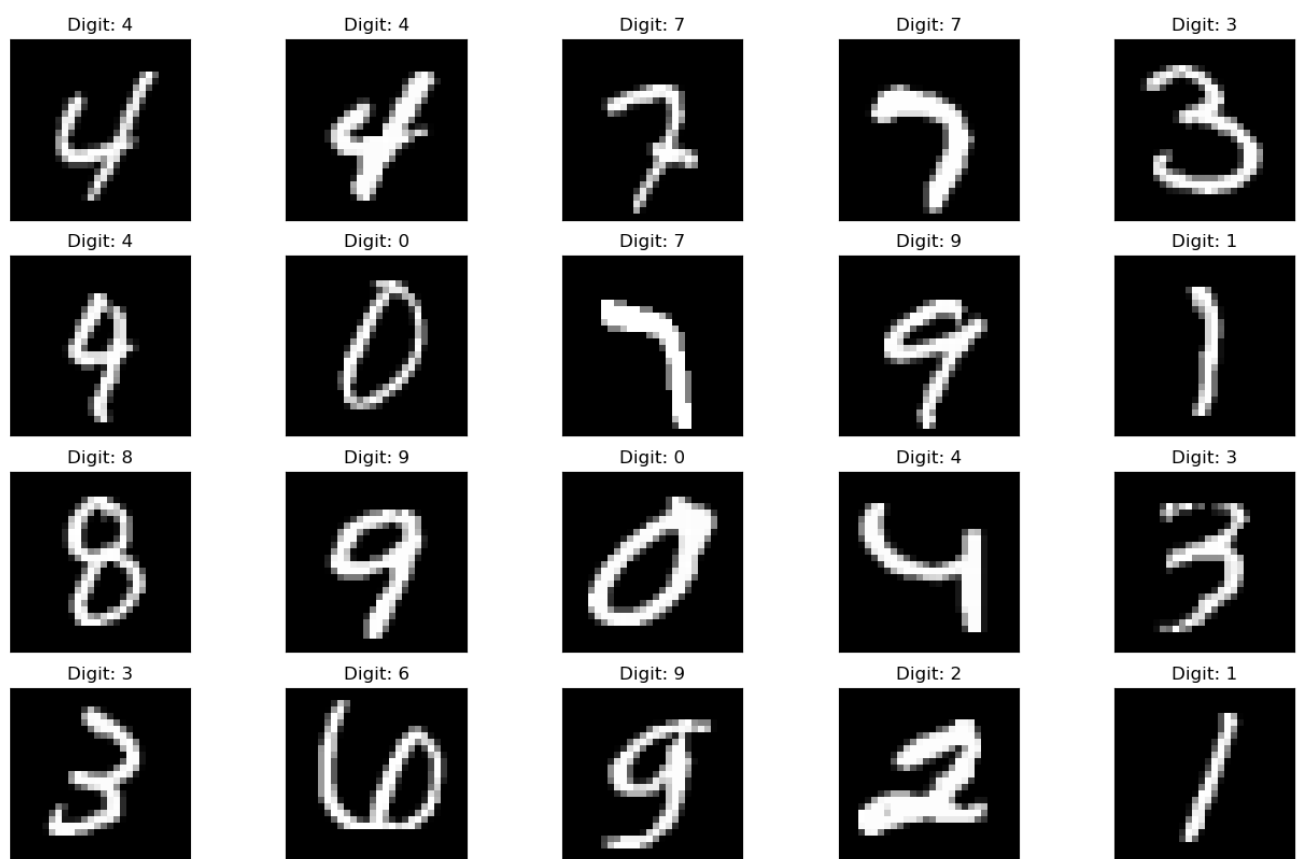
```
In [13]: predictions = np.argmax(model.predict(X_test), axis=-1)
         accuracy_score(y_test, predictions)
```

Out [13]: 0.9872

```
In [14]: n=random.randint(0,9999)
         plt.imshow(X_test[n])
         plt.show()
```



```
In [15]: predicted_value=model.predict(X_test)
         print("Handwritten number in the image is= %d" %np.argmax(predicted_value[n
```

Handwritten number in the image is= 0

```
In [16]: 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: 0.04624301567673683
Test accuracy: 0.9872000217437744

```
In [17]: #The implemented CNN model is giving Loss=0.04624301567673683  and
         #accuracy: 0.9872000217437744 for test mnist dataset
```