In [1]:

```
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
```

In [2]:

```
mnist=tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test)=mnist.load_data()
```

In [3]:

```
print(x_train.shape)
```

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

In [4]:

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

Out[4]:

```
(0, 255)
```

In [5]:

```
x_train = (x_train - 0.0) / (255.0 - 0.0)
x_test = (x_test - 0.0) / (255.0 - 0.0)
```

In [6]:

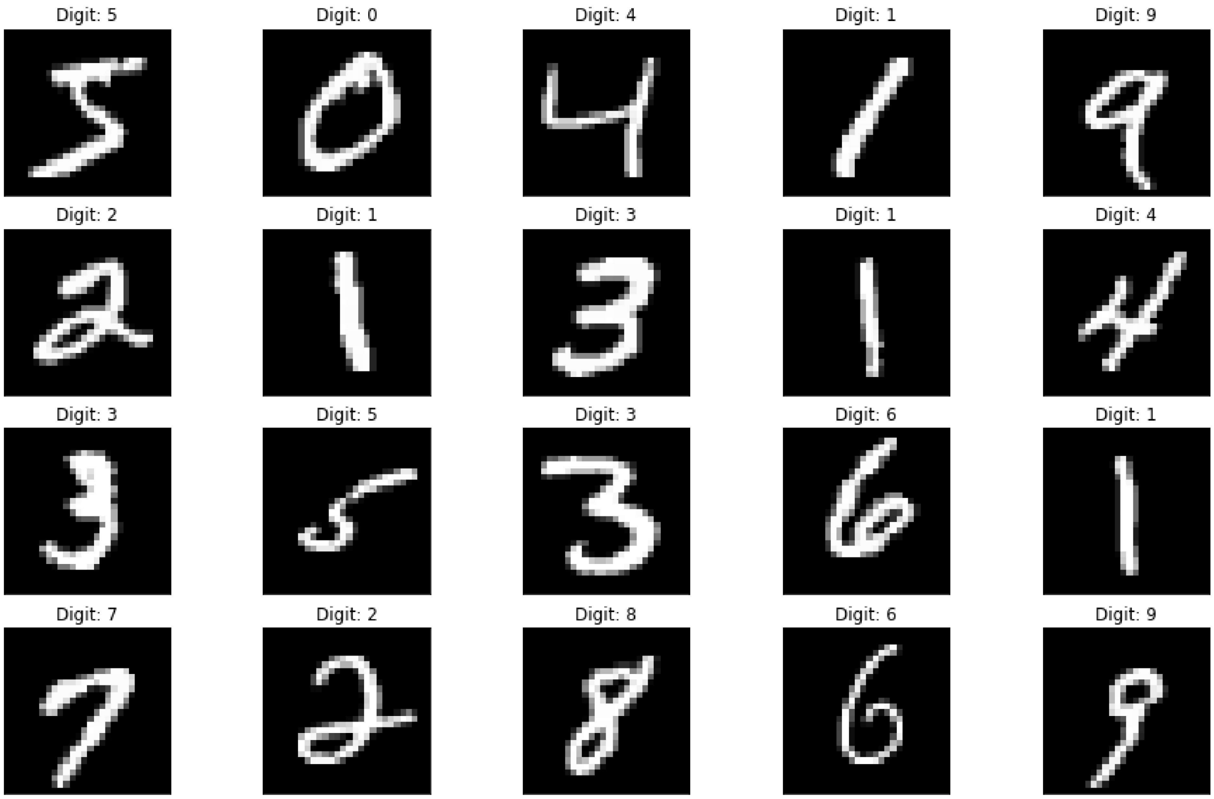
```
x_train[0].min(), x_train[0].max()
```

Out[6]:

```
(0.0, 1.0)
```

In [7]:

```
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 [8]:

```
x_train = x_train.reshape((x_train.shape + (1,)))
x_test = x_test.reshape((x_test.shape + (1,)))
```

In [9]:

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

Out[9]:

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

In [10]:

```
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 [11]:

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

In [12]:

```
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
_________________________________________________________________
```

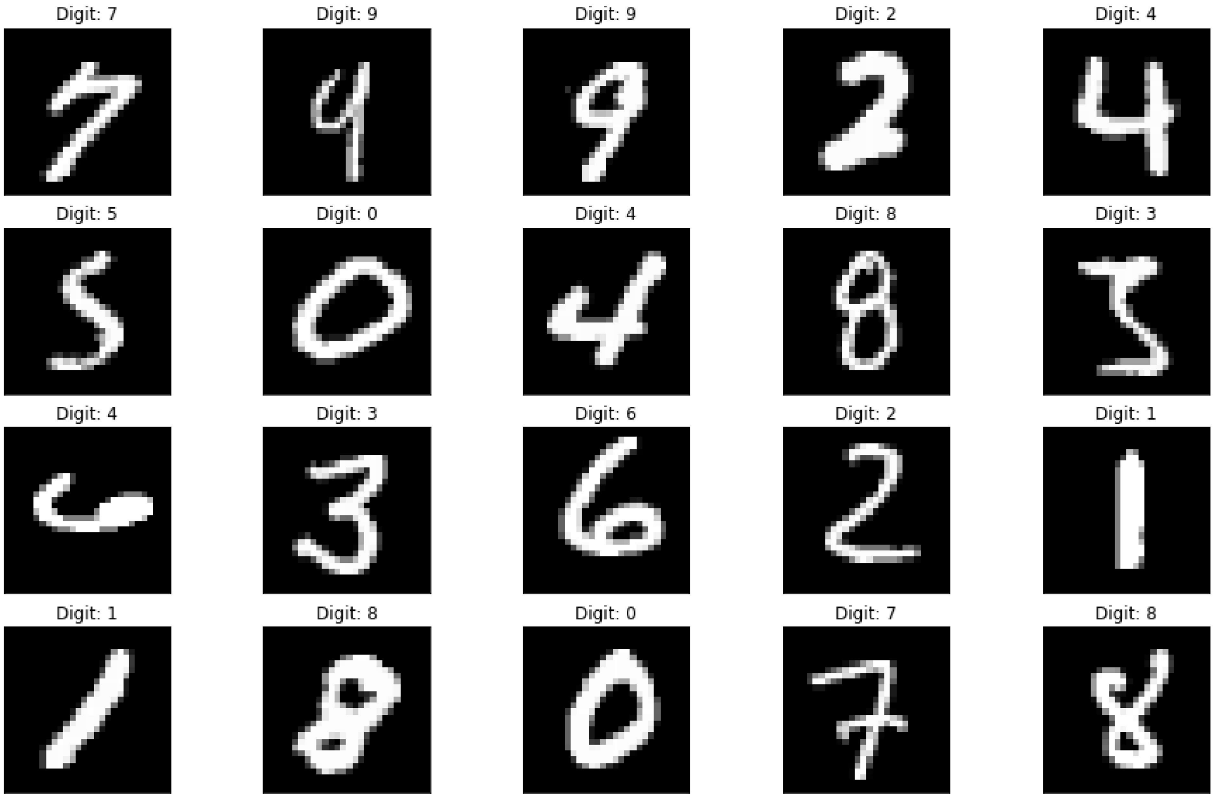
In [13]:

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

```
Epoch 1/10
1875/1875 [==============================] - 33s 17ms/step - loss: 0.2390 - acc
uracy: 0.9265
Epoch 2/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0764 - acc
uracy: 0.9769
Epoch 3/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0498 - acc
uracy: 0.9851
Epoch 4/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0354 - acc
uracy: 0.9891
Epoch 5/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0268 - acc
uracy: 0.9913
Epoch 6/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0204 - acc
uracy: 0.9939
Epoch 7/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0145 - acc
uracy: 0.9957
Epoch 8/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0106 - acc
uracy: 0.9971
Epoch 9/10
1875/1875 [==============================] - 36s 19ms/step - loss: 0.0077 - acc
uracy: 0.9981
Epoch 10/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0053 - acc
uracy: 0.9988
```

In [14]:

```
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()
```



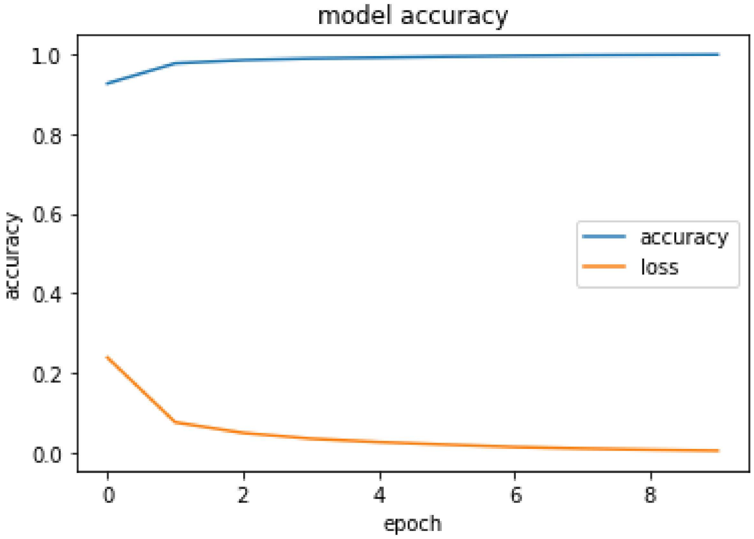
In [15]:
```
predictions = np.argmax(model.predict(x_test), axis=-1)
accuracy_score(y_test, predictions)
```

Out[15]:
```
0.9876
```

In [16]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.040274854749441 15
Test accuracy: 0.9876000285148621
```

In [25]:
```
plt.plot(model_log.history['accuracy'],label="accuracy")
plt.plot(model_log.history['loss'],label="loss")
plt.title('model accuracy')
plt.legend(['accuracy','loss'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [26]:

```
test_predict=model.predict(x_test)
test_predict_labels=np.argmax(test_predict,axis=1)
confusion_matrix=tf.math.confusion_matrix(labels=y_test,predictions=test_predict_
print("confusion matrix of the test set:\n",confusion_matrix)
```

```
confusion matrix of the test set:
 tf.Tensor(
[[ 973    0    1    0    0    0    1    1    3    1]
 [    0 1130    0    1    0    1    2    0    1    0]
 [    0    1 1017    2    2    0    1    6    2    1]
 [    0    0    1  999    0    2    0    3    2    3]
 [    0    0    1    0  973    0    0    0    1    7]
 [    1    0    0    6    0  882    1    0    1    1]
 [    5    4    2    1    3    1  939    0    3    0]
 [    0    2    4    1    0    0    0 1020    0    1]
 [    4    0    2    4    0    1    0    2  956    5]
 [    1    3    0    2    8    1    0    5    2  987]], shape=(10, 10), dtype=in
t32)
```

In [ ]: