In [1]:

```
#Build the Image classification model by dividing the model into following 4 stages:
#a. Loading and preprocessing the image data
#b. Defining the model’s architecture
#c. Training the model
#d. Estimating the model’s performance
```

In [2]:

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

```
C:\Users\Suraj\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version
>=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.26.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [3]:

```
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 [4]:

```
#a. Loading and preprocessing the image data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

In [5]:

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

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

In [6]:

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

Out[6]:

```
(0, 255)
```

In [7]:

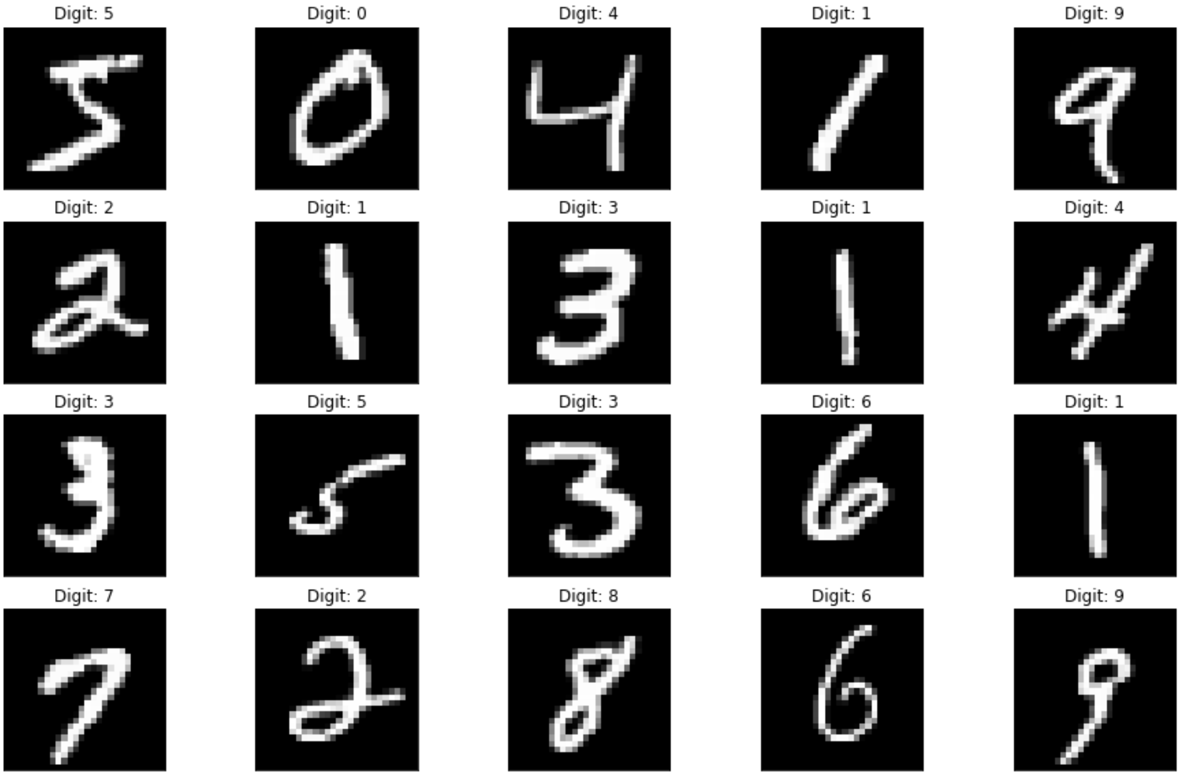
```
X_train = (X_train - 0.0) / (255.0 - 0.0)
X_test = (X_test - 0.0) / (255.0 - 0.0)
X_train[0].min(), X_train[0].max()
```

Out[7]:

```
(0.0, 1.0)
```

In [8]:

```
#b. Defining the model’s architecture
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]:

```
#c. Training the model
X_train = X_train.reshape((X_train.shape + (1,)))
X_test = X_test.reshape((X_test.shape + (1,)))
```

In [10]:

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

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]:

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

```
Epoch 1/10
1875/1875 [==============================] - 14s 7ms/step - loss: 0.2223 - accuracy: 0.932
9
Epoch 2/10
1875/1875 [==============================] - 14s 8ms/step - loss: 0.0683 - accuracy: 0.97
95
Epoch 3/10
1875/1875 [==============================] - 16s 8ms/step - loss: 0.0453 - accuracy: 0.98
64
Epoch 4/10
1875/1875 [==============================] - 17s 9ms/step - loss: 0.0335 - accuracy: 0.99
00
Epoch 5/10
1875/1875 [==============================] - 16s 9ms/step - loss: 0.0239 - accuracy: 0.99
26
Epoch 6/10
1875/1875 [==============================] - 16s 9ms/step - loss: 0.0176 - accuracy: 0.994
6
Epoch 7/10
1875/1875 [==============================] - 16s 9ms/step - loss: 0.0138 - accuracy: 0.99
57
Epoch 8/10
1875/1875 [==============================] - 16s 9ms/step - loss: 0.0095 - accuracy: 0.99
72
Epoch 9/10
1875/1875 [==============================] - 17s 9ms/step - loss: 0.0071 - accuracy: 0.998
3
Epoch 10/10
1875/1875 [==============================] - 16s 9ms/step - loss: 0.0049 - accuracy: 0.99
88
```

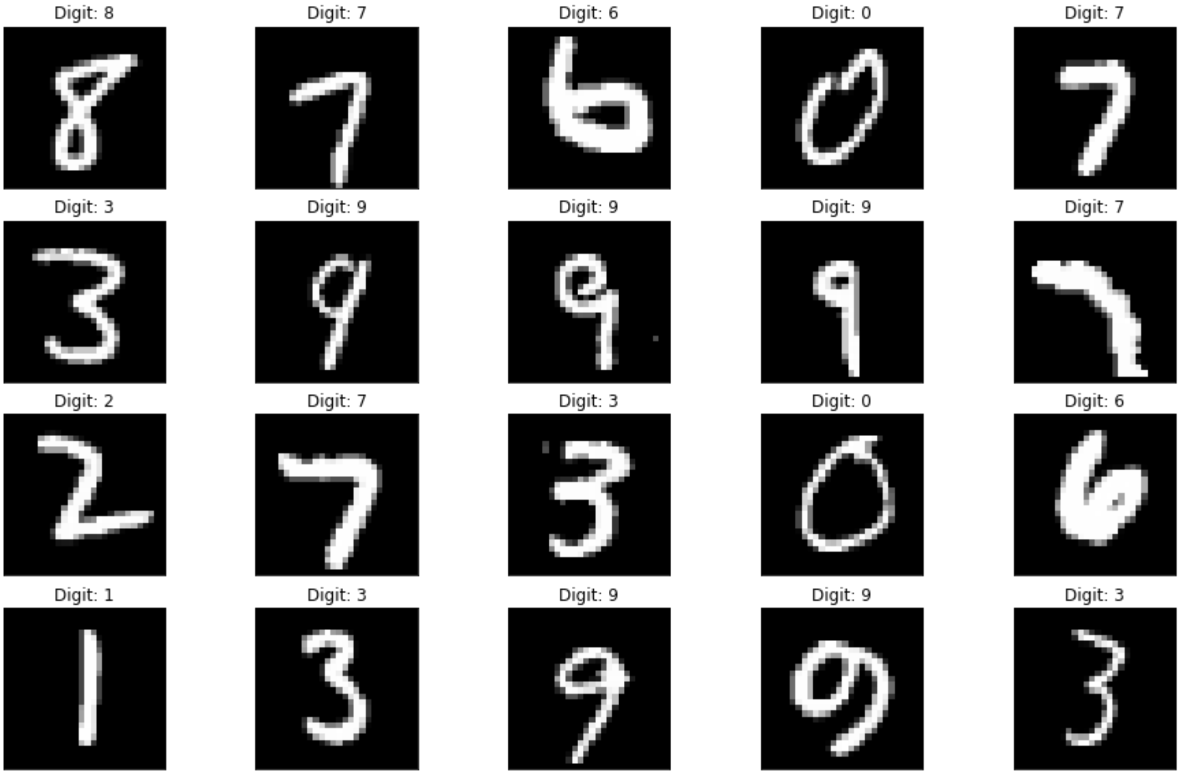
Out[14]:

```
<keras.src.callbacks.History at 0x1629e0c2eb0>
```

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 66ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
```



In [16]:

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

```
313/313 [==============================] - 1s 3ms/step
```

Out[16]:

```
0.9871
```

In [17]:

```
score=model.evaluate(X_test,y_test,verbose=0)
```

In [18]:

```
#d. Estimating the model's performance
print('Testloss:',score[0])
print('Test accuracy:', score[1])
```

```
Testloss: 0.044964201748371124
Test accuracy: 0.9871000051498413
```

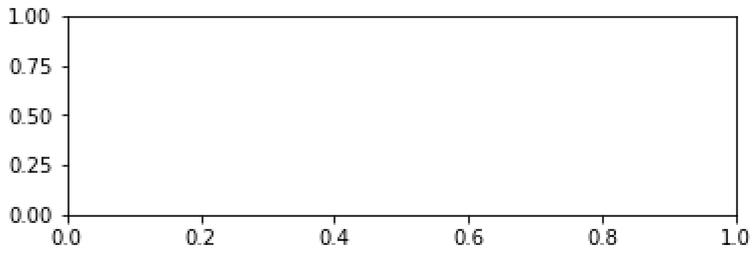
In [24]:

```
import os
# plotting the metrics
```

```
fig = plt.figure()
plt.subplot(2,1,1)
plt.plot(model_log.history['accuracy'])
plt.plot(model_log.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'],loc='lower right')
plt.subplot(2,1,2)
plt.plot(model_log.history['loss'])
plt.plot(model_log.history['val_loss'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [24], in <cell line: 6>()
      4 fig = plt.figure()
      5 plt.subplot(2,1,1)
----> 6 plt.plot(model_log.history['accuracy'])
      7 plt.plot(model_log.history['val_acc'])
      8 plt.title('model accuracy')

NameError: name 'model_log' is not defined
```

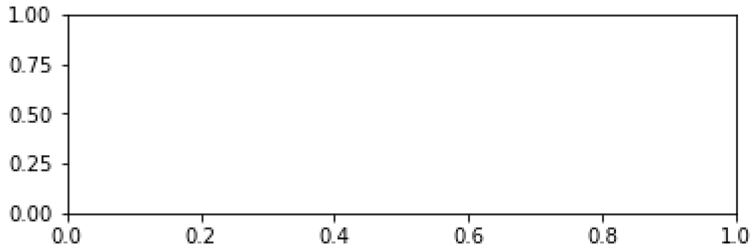


In [25]:

```
plt.subplot(211)
plt.plot(history['acc'])
plt.plot(history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='lower right')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [25], in <cell line: 2>()
      1 plt.subplot(211)
----> 2 plt.plot(history['acc'])
      3 plt.plot(history['val_acc'])
      4 plt.title('Model Accuracy')

NameError: name 'history' is not defined
```



In [ ]: