CNN - based model to classify MNIST data. Hand -written digits (0-9)

- Import the libraries
- Data reparation : Train-test split , specifying the shape of input.
- Build the CNN model
- Train and evaluate the model

In [5]:
```
# import the library
import numpy as np
import tensorflow as tf
from keras.datasets import mnist
from keras.layers import Dense, Dropout, Flatten, MaxPooling2D
from keras.layers import Conv2D
from keras import Sequential
import cv2
```

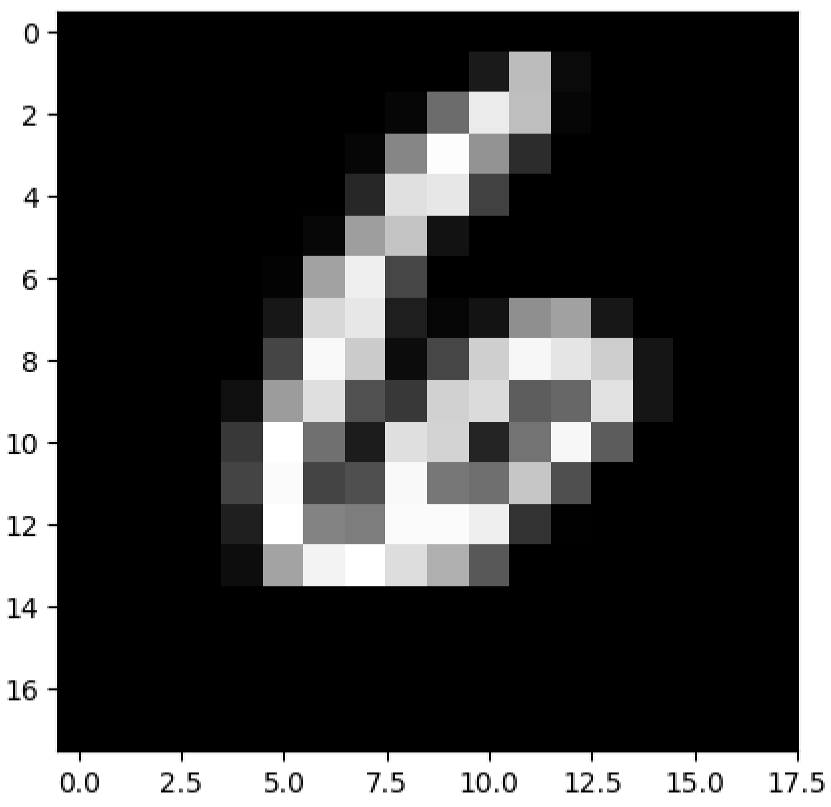
In [6]:
```
# load the MNISt dataset
(x_train, y_train) , (x_test , y_test ) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

In [7]:
```
# select one sample
x = x_train[13]
print('The shape of x is :', x.shape)
# resize the image
x = cv2.resize(x, (18,18))
```

```
The shape of x is : (28, 28)
```

In [8]:
```
# plot using matplotlib
import matplotlib.pyplot as plt
plt.imshow(x, cmap='gray')
plt.show()
```



In [9]:
```
print('The range of pixel vary from 0 to 255')
print(' Pixel value close to 0 is black and pixel value close to 255 is white')
print(x)
```

```
The range of pixel vary from 0 to 255
 Pixel value close to 0 is black and pixel value close to 255 is white
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  26 186  10   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   6 107 232 188   6   0   0   0   0   0]
 [  0   0   0   0   0   0   0   6 132 250 146  45   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  39 221 228  65   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   7 156 193  18   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   3 160 236  69   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0  23 213 228  30   5  19 141 159  23   0   0   0   0]
 [  0   0   0   0   0  68 246 200  12  69 204 244 226 202  21   0   0   0]
 [  0   0   0   0  15 154 220  79  57 206 216  92 102 222  21   0   0   0]
 [  0   0   0   0  56 252 111  28 220 208  36 114 244  91   0   0   0   0]
 [  0   0   0   0  67 248  67  78 246 118 110 195  78   0   0   0   0   0]
 [  0   0   0   0  31 252 129 124 248 248 236  50   1   0   0   0   0   0]
 [  0   0   0   0  13 162 240 252 218 173  87   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

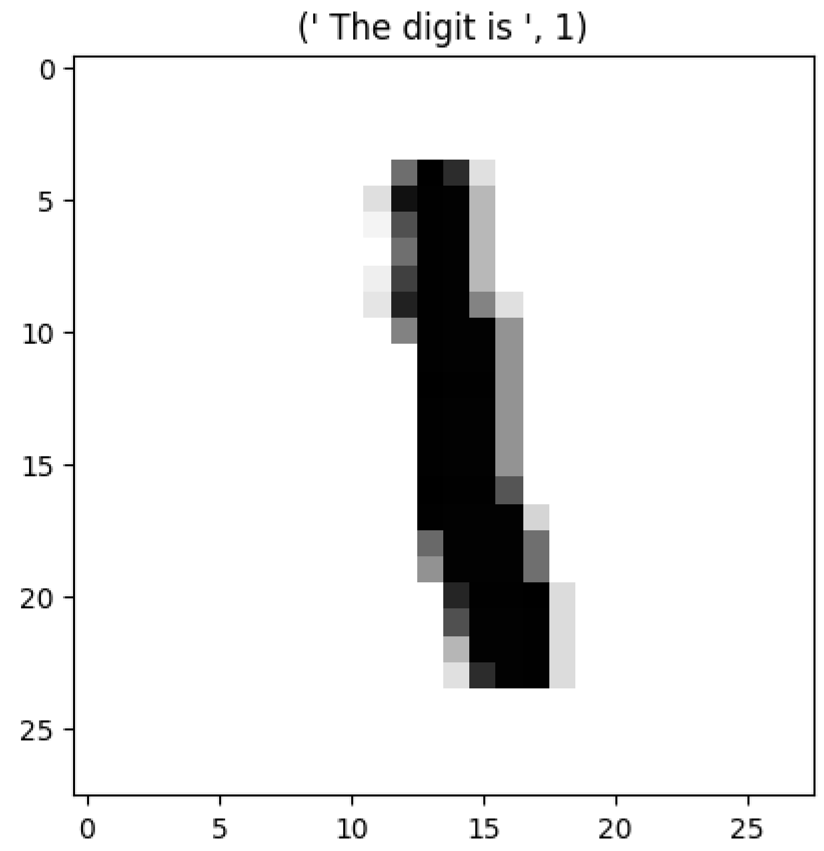
In [11]:
```
# shape of train and test
print("Shape of x-train data is :" , x_train.shape)
print("Shape of y-train data is :" , y_train.shape)
print("Shape of x-test data is :" , x_test.shape)
print("Shape of y-test data is :" , y_test.shape)
```

```
Shape of x-train data is : (60000, 28, 28)
Shape of y-train data is : (60000,)
Shape of x-test data is : (10000, 28, 28)
Shape of y-test data is : (10000,)
```

In [12]:
```
# sample 20,000 images
idx = np.random.randint(x.shape[0], size=20000)
x_train = x_train[idx,:]
y_train = y_train[idx]
print(x_train.shape)
print(y_train.shape)
```

```
(20000, 28, 28)
(20000,)
```

In [14]:
```
# print the digit
plt.imshow(x_train[13], cmap='binary')
label = y_train[13]
title1 = " The digit is ", label
plt.title(title1)
plt.show()
```



In [15]:
```
# specify the dimension of each image
img_rows , img_cols = 28,28
# batch size , number of classes , epoch
batch_size = 128
num_classes = 10
epoch = 12
```

In [17]:
```
#reshape train and test
x_train =x_train.reshape(x_train.shape[0],img_rows, img_cols , 1) # 1 channnel for grey scale image
x_test = x_test.reshape(x_test.shape[0],img_rows, img_cols , 1) # 1 channnel for grey scale image
print(x_train.shape)
print(x_test.shape)
```

```
(20000, 28, 28, 1)
(10000, 28, 28, 1)
```

In [18]:
```
# labels have to converted to categorical
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
print(y_train.shape)
print(y_test.shape)
```

```
(20000, 10)
(10000, 10)
```

In [21]:
```
y_train[2]
```

Out[21]:
```
array([0., 1., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```