In [ ]:

*## Name : Satyam kumar ## Roll no:I4250*

*## Subject:LP-IV(DL)*

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]:

(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) X\_train[0]**.**min(), X\_train[0]**.**max()

Out[5]: (0.0, 1.0)

In [6]:

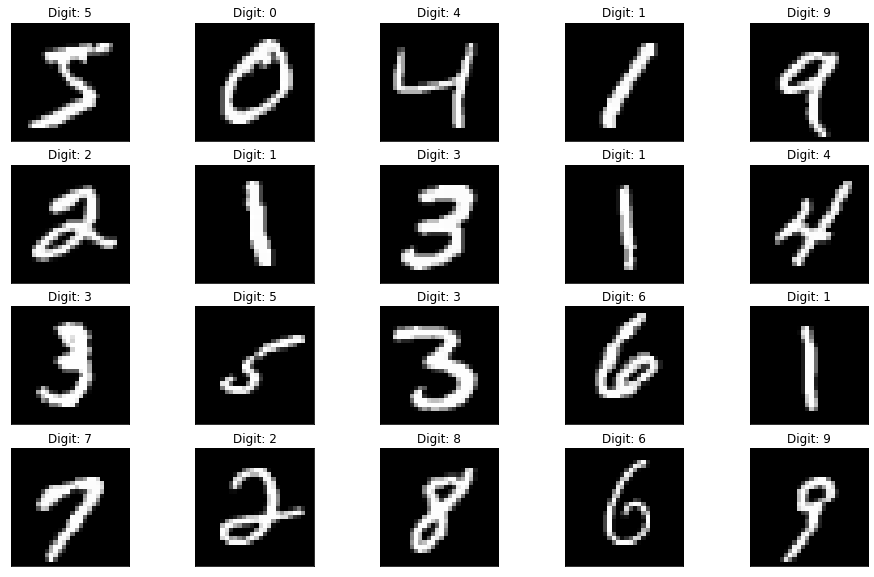
**def** plot\_digit(image, digit, plt, i): plt**.**subplot(4, 5, i **+** 1)

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]:

X\_train **=** X\_train**.**reshape((X\_train**.**shape **+** (1,))) X\_test **=** X\_test**.**reshape((X\_test**.**shape **+** (1,)))

In [8]:

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]:

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]:

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]:

model**.**fit(X\_train, y\_train, epochs**=**10, batch\_size**=**32)

Epoch 1/10

1875/1875 [==============================] - 19s 10ms/step - loss: 0.2392 - accuracy: 0.9275

Epoch 2/10

1875/1875 [==============================] - 18s 9ms/step - loss: 0.0801 - accuracy: 0.9757

Epoch 3/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0511 - accuracy: 0.9846

Epoch 4/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0352 - accuracy: 0.9893

Epoch 5/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0268 - accuracy: 0.9917

Epoch 6/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0203 - accuracy: 0.9938

Epoch 7/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0145 - accuracy: 0.9955

Epoch 8/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0107 - accuracy: 0.9969

Epoch 9/10

1875/1875 [==============================] - 19s 10ms/step - loss: 0.0092 - accuracy: 0.9975

Epoch 10/10

1875/1875 [==============================] - 18s 10ms/step - loss: 0.0066 - accuracy: 0.9983

Out[11]: <keras.callbacks.History at 0x27b921bdfa0>

In [12]:

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 [13]:

1/1 [==============================] - 0s 105ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 18ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 18ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 18ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 20ms/step

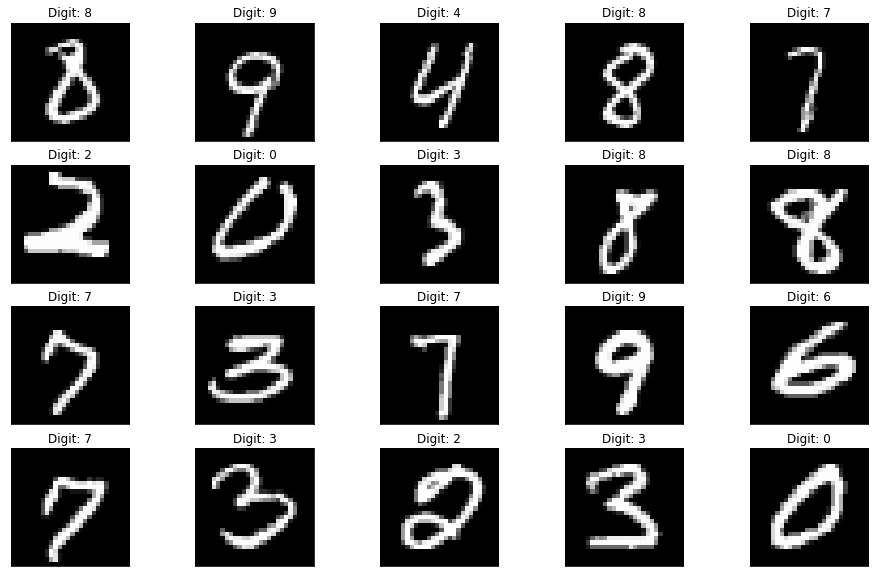
1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 21ms/step

1/1 [==============================] - 0s 20ms/step

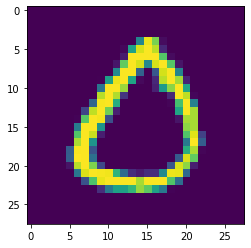


313/313 [==============================] - 1s 3ms/step Out[13]: 0.9881

predictions **=** np**.**argmax(model**.**predict(X\_test), axis**=-**1) accuracy\_score(y\_test, predictions)

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]))

In [16]:

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

Handwritten number in the image is= 0

score **=** model**.**evaluate(X\_test, y\_test, verbose**=**0) print('Test loss:', score[0]) *#Test loss: 0.0296396646054* print('Test accuracy:', score[1])

In [17]:

Test loss: 0.03916610777378082

Test accuracy: 0.988099992275238

*#The implemented CNN model is giving Loss=0.04624301567673683 and #accuracy: 0.9872000217437744 for test mnist dataset*