In[]:
```
##Name: Tanmay Gujar
##Rollno:I4159
##Subject:LP-IV(DL)
```

In[1]:
```
import numpy as
npimportpandasaspd
importrandom
importtensorflowastf
importmatplotlib.pyplotasplt
fromsklearn.metricsimportaccuracy_score

fromtensorflow.keras.modelsimportSequential
fromtensorflow.keras.layersimportFlatten,Conv2D,Dense,MaxPooling2D
fromtensorflow.keras.optimizersimportSGD
fromtensorflow.keras.utilsimportto_categorical
fromtensorflow.keras.datasetsimportmnist
```

In[2]:
```
(X_train,y_train),(X_test,y_test)=mnist.load_data()
```

In[3]:
```
print(X_train.shape)
```
```
(60000,28,28)
```

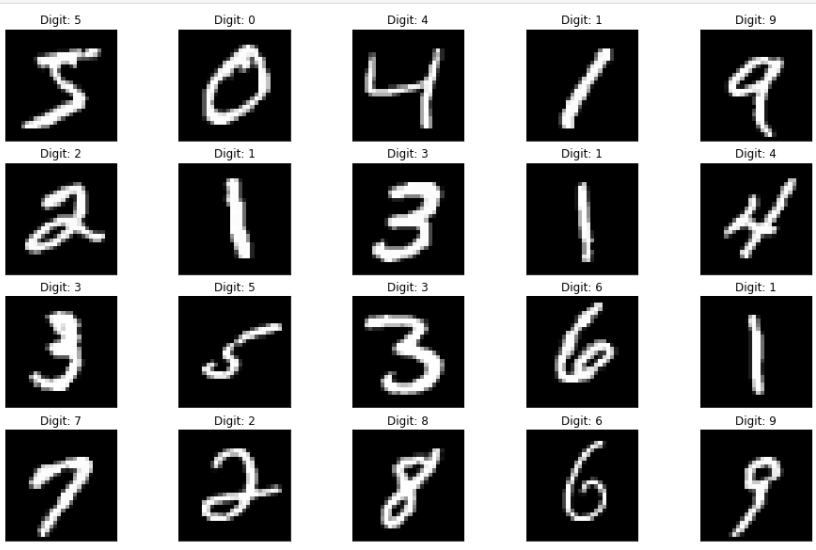
In[4]:
```
X_train[0].min(),X_train[0].max()
```

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]:
```
defplot_digit(image,digit,plt,i):plt
    .subplot(4,5,i+1)
    plt.imshow(image,cmap=plt.get_cmap('gray'))
    plt.title(f"Digit:{digit}")p
    lt.xticks([])
    plt.yticks([])plt.figure
(figsize=(16,10))foriinrange
(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_te
st=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,activat
    ion="softmax")
])
```

In[10]:
```
optimizer=SGD(learning_rate=0.01,momentum=0.9)mod
el.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",metr
    ics=["accuracy"]
)
model.summary()
```

```
Model:"sequential"
_________________________________________________________________
 Layer(type)                 OutputShape              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

=================================================================
Totalparams:542,230
Trainableparams:542,230
Non-trainableparams:0
_________________________________________________________________
```

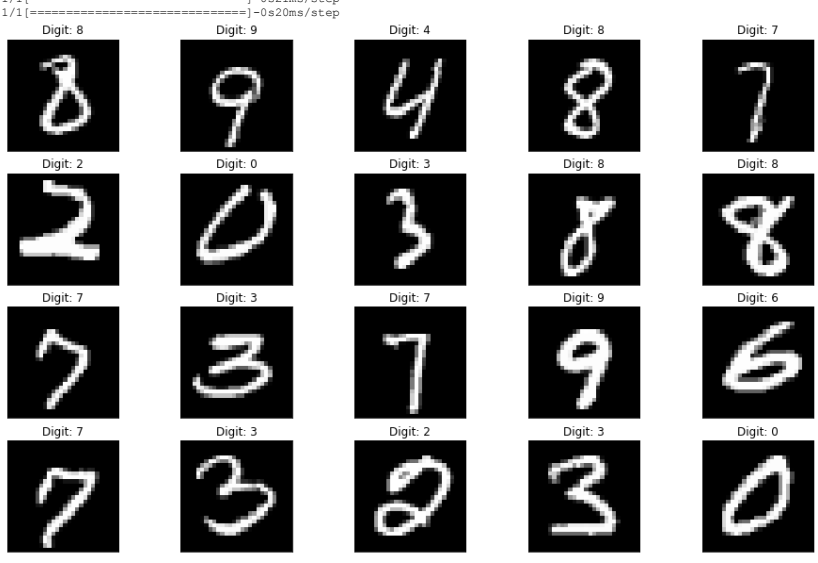
In[11]:
```
model.fit(X_train,y_train,epochs=10,batch_size=32)
```

```
Epoch1/10
1875/1875[==============================]-19s10ms/step-loss:0.2392-accuracy:0.9275
Epoch2/10
1875/1875[==============================]-18s9ms/step-loss:0.0801-accuracy:0.9757
Epoch3/10
1875/1875[==============================]-18s10ms/step-loss:0.0511-accuracy:0.9846
Epoch4/10
1875/1875[==============================]-18s10ms/step-loss:0.0352-accuracy:0.9893
Epoch5/10
1875/1875[==============================]-18s10ms/step-loss:0.0268-accuracy:0.9917
Epoch6/10
1875/1875[==============================]-18s10ms/step-loss:0.0203-accuracy:0.9938
Epoch7/10
1875/1875[==============================]-18s10ms/step-loss:0.0145-accuracy:0.9955
Epoch8/10
1875/1875[==============================]-18s10ms/step-loss:0.0107-accuracy:0.9969
Epoch9/10
1875/1875[==============================]-19s10ms/step-loss:0.0092-accuracy:0.9975
Epoch10/10
1875/1875[==============================]-18s10ms/step-loss:0.0066-accuracy:0.9983
```

Out[11]:<keras.callbacks.Historyat0x27b921bdfa0>

In[12]:
```
plt.figure(figsize=(16,10))
foriinrange(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()
```

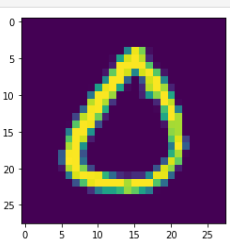
```
1/1[==============================]-0s105ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s18ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s18ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s18ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s19ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s20ms/step
1/1[==============================]-0s21ms/step
1/1[==============================]-0s20ms/step
```



In[13]:
```
predictions=np.argmax(model.predict(X_test),axis=-
1)accuracy_score(y_test,predictions)
```

```
313/313[==============================]-
```
1s3ms/stepOut[13]:0.9881

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



In[15]:
```
predicted_value=model.predict(X_test)
print("Handwrittennumberintheimageis=%d"%np.argmax(predicted_value[n]))
```

```
313/313[==============================]-1s3ms/step
Handwrittennumberintheimageis=0
```

In[16]:
```
score = model.evaluate(X_test, y_test,
verbose=0)print('Testloss:',score[0])#Testloss:0.0296396
646054print('Testaccuracy:',score[1])
```

```
Testloss:0.039166107773780082
Testaccuracy:0.988099992275238
```

In[17]:
```
#TheimplementedCNNmodelisgivingLoss=0.04624301567673683and#accura
cy:0.98720002174377744fortestmnistdataset
```