SPRINT1-UNDERSTANDING THE DATA

1.Import the required libraries

```
import numpy
import tensorflow
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense ,Flatten
from tensorflow.keras.layers import Conv2D
from keras.optimizers import Adam
from keras.utils import np_utils
```

2.Loading the data

```
(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
```

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

```
(60000, 28, 28)
(10000, 28, 28)
```

3.Analyzing the data

```
x_train[0]
```

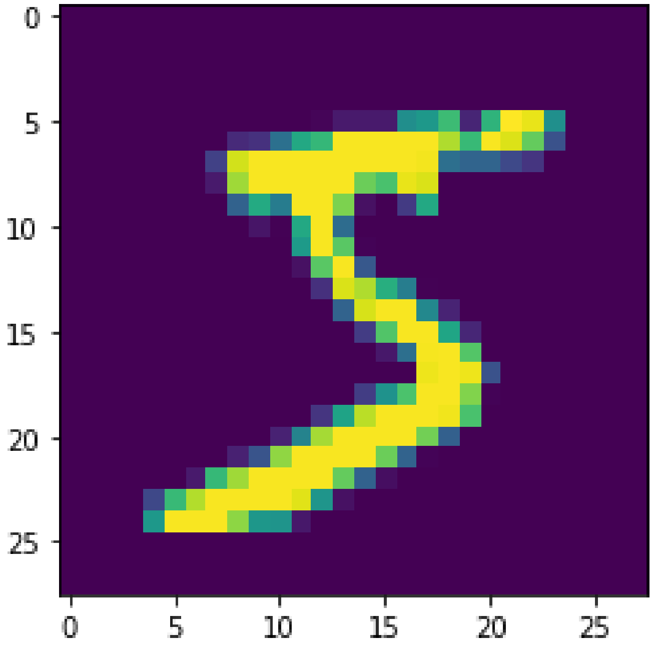
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   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,   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,
3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154,
170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253,
253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253,
253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253,
253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154,
253,
         90,   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, 139,
253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11,
190,
        253,  70,   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,
35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0,  16,  93, 252, 253, 187,   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, 249, 253, 249,  64,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114,
221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253,
253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253,
253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244,
133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,
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,   0,   0,   0,   0,   0,
0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0,
          0,   0]], dtype=uint8)
```

```
y_train[0]
```

```
5
```

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7f0428b69bd0>
```



4.Reshaping the dataset

```
x_train=x_train.reshape(60000, 28, 28, 1).astype('float32')
x_test=x_test.reshape(10000, 28, 28, 1).astype('float32')
```

5.One hot Encoding

```
number_of_classes=10
y_train=np_utils.to_categorical(y_train, number_of_classes)
y_test=np_utils.to_categorical(y_test, number_of_classes)
```

```
y_train[0]
```

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

SPRINT2-MODEL BUILDING

1.Add CNN Layers

```
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(28,28,1),activation='relu'))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(Flatten())
model.add(Dense(number_of_classes,activation='softmax'))
```

2.Compiling the model

```
model.compile(loss='categorical_crossentropy',optimizer="Adam",metrics
=['accuracy'])
```

3.Train the model

```
model.fit(x_train,y_train,
validation_data=(x_test,y_test),epochs=5,batch_size=32)
```

```
Epoch 1/5
1875/1875 [==============================] - 196s 104ms/step - loss:
0.2613 - accuracy: 0.9523 - val_loss: 0.0924 - val_accuracy: 0.9719
Epoch 2/5
1875/1875 [==============================] - 194s 103ms/step - loss:
0.0669 - accuracy: 0.9802 - val_loss: 0.0714 - val_accuracy: 0.9790
Epoch 3/5
1875/1875 [==============================] - 194s 104ms/step - loss:
0.0416 - accuracy: 0.9872 - val_loss: 0.1121 - val_accuracy: 0.9702
Epoch 4/5
1875/1875 [==============================] - 194s 103ms/step - loss:
0.0321 - accuracy: 0.9900 - val_loss: 0.0774 - val_accuracy: 0.9819
Epoch 5/5
1875/1875 [==============================] - 193s 103ms/step - loss:
0.0276 - accuracy: 0.9914 - val_loss: 0.0944 - val_accuracy: 0.9776
```

```
<keras.callbacks.History at 0x7f04225fd910>
```

4.Observing the metics

```
metrics=model.evaluate(x_test,y_test,verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy): 
[0.09440457075834274, 0.9775999784469604]
```

5.Test the model

```
prediction=model.predict(x_test[:4])
print(prediction)
```

```
1/1 [==============================] - 1s 503ms/step
[[6.39302014e-11 1.92686877e-19 6.10958750e-08 4.46125068e-06
  5.08821697e-19 2.58780797e-14 5.66186983e-19 9.99995470e-01
  9.95120999e-12 4.52038990e-10]
 [5.31988949e-13 2.69534097e-13 1.00000000e+00 2.39036747e-14
  5.06575230e-20 4.59198028e-22 1.59037870e-11 2.60662725e-23
  7.69959183e-13 4.27862457e-24]
 [5.10552134e-08 9.99419808e-01 1.03479006e-05 7.75968352e-12
  1.03664997e-05 3.92677890e-07 1.58789104e-09 3.52089885e-11
  5.59113978e-04 1.09291005e-11]
 [1.00000000e+00 5.53433434e-16 4.46223137e-12 2.54150012e-15
  3.18195550e-14 3.07361664e-10 7.02960912e-10 2.14802486e-16
  2.28522984e-13 7.06658430e-12]]
```

```
import numpy as np
print(np.argmax(prediction,axis=1))
print(y_test[:4])
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

6.Observing the metics

```
metrics=model.evaluate(x_test,y_test,verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy): 
[0.09440457075834274, 0.9775999784469604]
```

7.Test the model

```
prediction=model.predict(x_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 23ms/step
[[6.39302014e-11 1.92686877e-19 6.10958750e-08 4.46125068e-06
  5.08821697e-19 2.58780797e-14 5.66186983e-19 9.99995470e-01
  9.95120999e-12 4.52038990e-10]
 [5.31988949e-13 2.69534097e-13 1.00000000e+00 2.39036747e-14
  5.06575230e-20 4.59198028e-22 1.59037870e-11 2.60662725e-23
  7.69959183e-13 4.27862457e-24]
 [5.10552134e-08 9.99419808e-01 1.03479006e-05 7.75968352e-12
  1.03664997e-05 3.92677890e-07 1.58789104e-09 3.52089885e-11
  5.59113978e-04 1.09291005e-11]
 [1.00000000e+00 5.53433434e-16 4.46223137e-12 2.54150012e-15
  3.18195550e-14 3.07361664e-10 7.02960912e-10 2.14802486e-16
  2.28522984e-13 7.06658430e-12]]
```

```
import numpy as np
print(np.argmax(prediction,axis=1))
print(y_test[:4])
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

8.Save the model

```
model.save('models/mnistCNN.h5')
```

9.Test with saved model

Testing-1

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

model=load_model('/content/models/mnistCNN.h5')
img=image.load_img('/content/data.jpg',target_size=(28,28),grayscale=T
rue)
img=image.img_to_array(img)
print(img.shape)
x=np.expand_dims(img,axis=0)
print(x.shape)
print('*'*20)
print(model.predict(x))
print('*'*20)
print(np.round_(model.predict(x)))
```

```
(28, 28, 1)
(1, 28, 28, 1)
********************
1/1 [==============================] - 0s 58ms/step
```

```
[[3.7480348e-03 7.6842160e-10 3.1971972e-06 1.1066861e-03 8.1795186e-
04
  2.6664600e-01 3.0438601e-05 1.0285763e-02 5.6085092e-01 1.5651101e-
01]]
********************
1/1 [==============================] - 0s 21ms/step
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
```

Testing-2

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

model=load_model('/content/models/mnistCNN.h5')
img=image.load_img('/content/0.png',target_size=(28,28),grayscale=True
)
img=image.img_to_array(img)
print(img.shape)
x=np.expand_dims(img,axis=0)
print(x.shape)
print('*'*20)
print(model.predict(x))
print('*'*20)
print(np.round_(model.predict(x)))
```

```
(28, 28, 1)
(1, 28, 28, 1)
********************
1/1 [==============================] - 0s 58ms/step
[[8.4565616e-01 6.1223082e-06 1.3529530e-01 7.2525195e-06 7.0594538e-
06
  7.8179939e-03 6.0109678e-06 1.0849683e-02 1.4368325e-06 3.5300280e-
04]]
********************
1/1 [==============================] - 0s 34ms/step
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Testing-3

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

model=load_model('/content/models/mnistCNN.h5')
img=image.load_img('/content/5.png',target_size=(28,28),grayscale=True
)
img=image.img_to_array(img)
print(img.shape)
x=np.expand_dims(img,axis=0)
print(x.shape)
print('*'*20)
print(model.predict(x))
print('*'*20)
print(np.round_(model.predict(x)))
```

```
WARNING:tensorflow:5 out of the last 9 calls to <function
Model.make_predict_function.<locals>.predict_function at
0x7f041e341f80> triggered tf.function retracing. Tracing is expensive
and the excessive number of tracings could be due to (1) creating
@tf.function repeatedly in a loop, (2) passing tensors with different
shapes, (3) passing Python objects instead of tensors. For (1), please
define your @tf.function outside of the loop. For (2), @tf.function
has reduce_retracing=True option that can avoid unnecessary retracing.
For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more
details.
```

```
(28, 28, 1)
(1, 28, 28, 1)
********************
1/1 [==============================] - 0s 60ms/step
[[1.00980415e-05 1.88434527e-07 6.88326254e-05 1.65369492e-02
  6.14392848e-05 9.82516408e-01 3.10022413e-04 2.06910133e-08
  4.71084146e-04 2.49948462e-05]]
********************
1/1 [==============================] - 0s 23ms/step
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
```