```
import tensorflow as tf
from tensorflow import keras
import  matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
plt.figure(figsize=(10,7))
```

```
<Figure size 720x504 with 0 Axes>
<Figure size 720x504 with 0 Axes>
```

```
(x_train,y_train),(x_test,y_test) = keras.datasets.mnist.load_data()
```

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

```
print("No Of Records in x_train : {} and y_train : {}".format(len(x_train),len(y_train)))
```

```
No Of Records in x_train : 60000 and y_train : 60000
```

```
print("No Of Records in x_test : {} and y_test : {}".format(len(x_test),len(y_test)))
```

```
No Of Records in x_test : 10000 and y_test : 10000
```

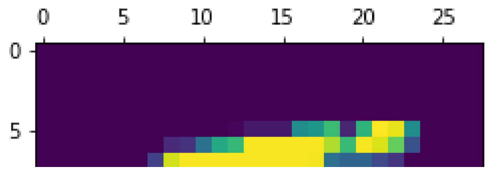
```
x_train[0]
```

```
          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)
```

Let's See The Values what are They : in X_train and Y_train first Element

```
plt.matshow(x_train[0])
plt.show()
```



```
y_train[0]
```

```
5
```

Shape Changing

```
x_train.shape
```

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

```
x_train[0].shape
```

```
(28, 28)
```

For IMage is 28 * 28 Format We have Flatten mean 28 * 28 = 784

```
28*28
```

```
784
```

```
# len(x_train) = 60000, 28*28 = 784 It mean 60000 Rows and 780 columns
x_train_flattend = x_train.reshape(len(x_train),28*28)
```

```
x_train_flattend
```

```
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]], dtype=uint8)
```

```
x_train_flattend.shape
```

```
(60000, 784)
```

```
# Next We Have to Flattend X_test
# len(x_train) = 10000, 28*28 = 784 It mean 10000 Rows and 780 columns
x_test_flattend = x_test.reshape(len(x_test),28*28)
```

```
x_test_flattend.shape
```

```
    (10000, 784)
```

Now we are going to create simple Neural Network with hidden layer first later we will add hidden layer. we have 784 input and number 0 - 9 so 10 output neurons

Sequential mean stack of layers in the networks „Since it is stack so it accept every as one element

keras.Sequential([keras.layers.Dense(10,input_shape=(784,),activation='sigmoid')])

```
model·=·keras.Sequential(
····[·keras.layers.Dense(10,input_shape=(784,),activation='sigmoid')]
)
model.compile(optimizer="adam",
··············loss='sparse_categorical_crossentropy',
··············metrics=['accuracy'])
model.fit(x_train_flattend,y_train,epochs=5)
```

```
    Epoch 1/5
    1875/1875 [==============================] - 4s 2ms/step - loss: 9.8163 - accuracy: 0.83
    Epoch 2/5
    1875/1875 [==============================] - 3s 2ms/step - loss: 6.1112 - accuracy: 0.87
    Epoch 3/5
    1875/1875 [==============================] - 3s 2ms/step - loss: 5.6685 - accuracy: 0.88
    Epoch 4/5
    1875/1875 [==============================] - 3s 2ms/step - loss: 5.4352 - accuracy: 0.88
    Epoch 5/5
    1875/1875 [==============================] - 3s 2ms/step - loss: 5.3695 - accuracy: 0.88
    <keras.callbacks.History at 0x7f711bea6b10>
```

By Scaling Let's Check theScore and We can observe the value between the 0 to 253 in x_train[0]

```
(x_trainS,y_trainS),(x_testS,y_testS) = keras.datasets.mnist.load_data()
```

```
x_trainS = x_trainS / 253
x_testS = x_testS / 253
```

```
x_trainS[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.        , 0.01185771, 0.07114625, 0.07114625,
        0.07114625, 0.49802372, 0.53754941, 0.6916996 , 0.1027668 ,
        0.65612648, 1.00790514, 0.97628458, 0.50197628, 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.11857708, 0.14229249,
        0.3715415 , 0.60869565, 0.67193676, 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.88932806, 0.6798419 ,
        1.        , 0.95652174, 0.77075099, 0.25296443, 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.19367589, 0.94071146, 1.        ,
        1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 0.99209486, 0.36758893, 0.32411067,
        0.32411067, 0.22134387, 0.1541502 , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.07114625, 0.86561265, 1.        ,
        1.        , 1.        , 1.        , 1.        , 0.7826087 ,
        0.71936759, 0.97628458, 0.95256917, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.31620553, 0.61660079,
            0.4229249 , 1.        , 1.        , 0.81027668, 0.04347826,
```

```
x_train_flattendS = x_train.reshape(len(x_trainS),28*28)
x_test_flattendS = x_test.reshape(len(x_testS),28*28)
```

```
modelS = keras.Sequential(
    [ keras.layers.Dense(10,input_shape=(784,),activation='sigmoid')]
)
modelS.compile(optimizer="adam",
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
modelS.fit(x_train_flattendS,y_trainS,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 9.5290 - accuracy: 0.84
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 6.1882 - accuracy: 0.87
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 5.8402 - accuracy: 0.88
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 5.5791 - accuracy: 0.88
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 5.4398 - accuracy: 0.88
<keras.callbacks.History at 0x7f71160f8a50>
```

WithOut Scaling is Good So we took with scaling for remaining process

```
model.evaluate(x_test_flattend,y_test)
```

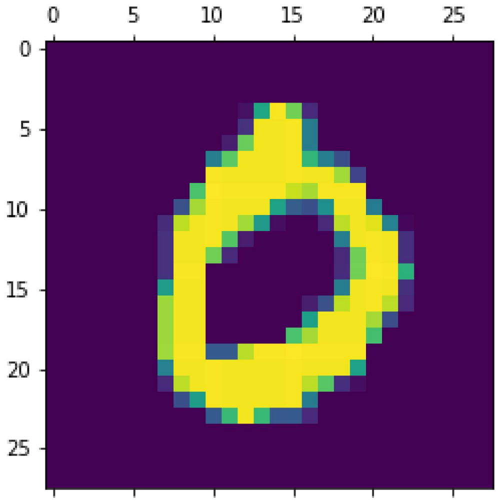
```
313/313 [==============================] - 1s 2ms/step - loss: 6.6010 - accuracy: 0.8722
[6.601004123687744, 0.8722000122070312]
```

With Visual

```
predicted = model.predict(x_test_flattend)
predicted
```

```
313/313 [==============================] - 1s 2ms/step
array([[0.000000e+00, 0.000000e+00, 9.955793e-01, ..., 1.000000e+00,
        1.000000e+00, 1.000000e+00],
       [1.000000e+00, 1.000000e+00, 1.000000e+00, ..., 0.000000e+00,
        1.000000e+00, 0.000000e+00],
       [0.000000e+00, 1.000000e+00, 1.000000e+00, ..., 9.904126e-01,
        9.998844e-01, 6.601462e-11],
       ...,
       [0.000000e+00, 0.000000e+00, 0.000000e+00, ..., 1.000000e+00,
        1.000000e+00, 1.000000e+00],
       [0.000000e+00, 0.000000e+00, 0.000000e+00, ..., 7.946653e-36,
        1.000000e+00, 0.000000e+00],
       [0.000000e+00, 0.000000e+00, 1.000000e+00, ..., 0.000000e+00,
        0.000000e+00, 0.000000e+00]], dtype=float32)
```

```
plt.matshow(x_test[3])
plt.show()
```



```
predicted[3]
```

```
    array([1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 4.9476516e-34,
           0.0000000e+00, 4.6334796e-26, 1.2247935e-11, 1.0000000e+00,
           5.1297760e-10, 9.9998993e-01], dtype=float32)
```

```
np.argmax(predicted[3])
```

```
    0
```

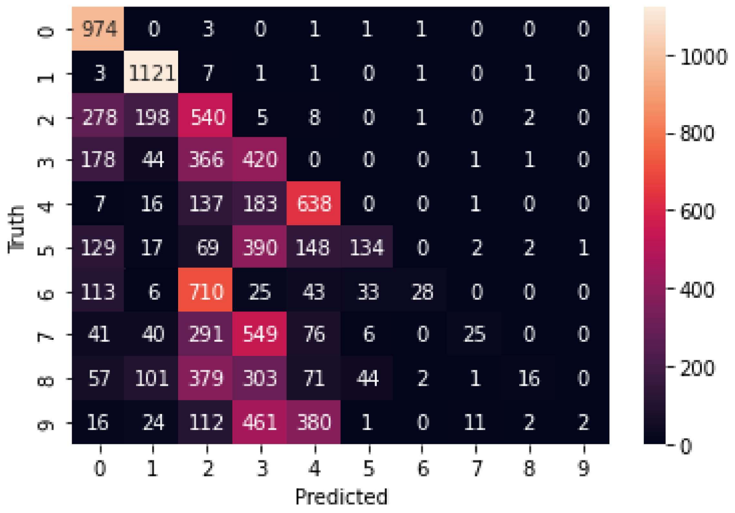
```
y_predicted_labels = [np.argmax(i) for i in predicted]
```

```
conf = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
conf
```

```
    <tf.Tensor: shape=(10, 10), dtype=int32, numpy=
    array([[ 974,    0,    3,    0,    1,    1,    1,    0,    0,    0],
           [   3, 1121,    7,    1,    1,    0,    1,    0,    1,    0],
           [ 278,  198,  540,    5,    8,    0,    1,    0,    2,    0],
           [ 178,   44,  366,  420,    0,    0,    0,    1,    1,    0],
           [   7,   16,  137,  183,  638,    0,    0,    1,    0,    0],
           [ 129,   17,   69,  390,  148,  134,    0,    2,    2,    1],
           [ 113,    6,  710,   25,   43,   33,   28,    0,    0,    0],
           [  41,   40,  291,  549,   76,    6,    0,   25,    0,    0],
           [  57,  101,  379,  303,   71,   44,    2,    1,   16,    0],
         [  16,    24,   112,   461,   380,     1,     0,    11,     2,     2]],
        dtype=int32)>
```

```
sns.heatmap(conf,annot=True,fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

```
    Text(33.0, 0.5, 'Truth')
```



Adding Hidden Layer

```
model = keras.Sequential(
    [ keras.layers.Dense(100,input_shape=(784,),activation='relu'),
     keras.layers.Dense(10,activation='sigmoid')]
)
model.compile(optimizer="adam",
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train_flattend,y_train,epochs=5)
```

```
    Epoch 1/5
    1875/1875 [==============================] - 6s 3ms/step - loss: 2.2457 - accuracy: 0.83
    Epoch 2/5
    1875/1875 [==============================] - 6s 3ms/step - loss: 0.4044 - accuracy: 0.96
    Epoch 3/5
    1875/1875 [==============================] - 5s 3ms/step - loss: 0.2927 - accuracy: 0.92
    Epoch 4/5
    1875/1875 [==============================] - 8s 4ms/step - loss: 0.2561 - accuracy: 0.93
    Epoch 5/5
    1875/1875 [==============================] - 8s 4ms/step - loss: 0.2246 - accuracy: 0.94
    <keras.callbacks.History at 0x7f7111ef0e90>
```

```
predicted1 = model.predict(x_test_flattend)
```

```
predicted1
```

```
313/313 [==============================] - 1s 4ms/step
array([[1.3854158e-11, 1.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0000000e+00],
       [1.0000000e+00, 1.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 9.8391801e-01],
       [6.6929517e-10, 1.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 9.9999583e-01],
       ...,
       [9.7029265e-28, 1.0000000e+00, 2.3444231e-04, ..., 1.0000000e+00,
        5.4457563e-01, 1.0000000e+00],
       [1.0000000e+00, 1.5913882e-16, 1.8500354e-05, ..., 1.0000000e+00,
        1.0000000e+00, 1.0000000e+00],
       [1.0000000e+00, 1.0000000e+00, 1.0000000e+00, ..., 4.2602935e-01,
        1.0000000e+00, 9.9999517e-01]], dtype=float32)
```

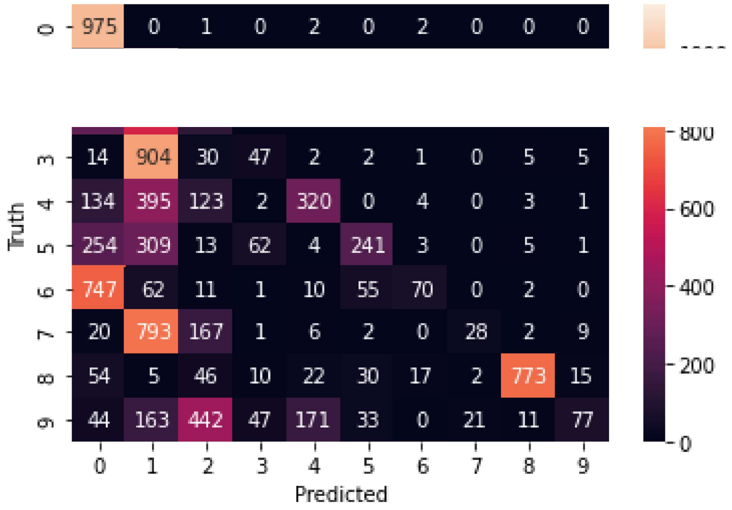
```
y_predicted_labels = [np.argmax(i) for i in predicted1]
```

```
conf = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
conf
```

```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 975,    0,    1,    0,    2,    0,    2,    0,    0,    0],
       [   2, 1123,    3,    1,    1,    0,    0,    0,    5,    0],
       [ 279,  600,  133,    1,    4,    0,    1,    1,   13,    0],
       [  14,  904,   30,   47,    2,    2,    1,    0,    5,    5],
       [ 134,  395,  123,    2,  320,    0,    4,    0,    3,    1],
       [ 254,  309,   13,   62,    4,  241,    3,    0,    5,    1],
       [ 747,   62,   11,    1,   10,   55,   70,    0,    2,    0],
       [  20,  793,  167,    1,    6,    2,    0,   28,    2,    9],
       [  54,    5,   46,   10,   22,   30,   17,    2,  773,   15],
       [  44,  163,  442,   47,  171,   33,    0,   21,   11,   77]],
      dtype=int32)>
```

```
sns.heatmap(conf,annot=True,fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

```
Text(33.0, 0.5, 'Truth')
```



	0	1	2	3	4	5	6	7	8	9
0	975	0	1	0	2	0	2	0	0	0
3	14	904	30	47	2	2	1	0	5	5
4	134	395	123	2	320	0	4	0	3	1
5	254	309	13	62	4	241	3	0	5	1
6	747	62	11	1	10	55	70	0	2	0
7	20	793	167	1	6	2	0	28	2	9
8	54	5	46	10	22	30	17	2	773	15
9	44	163	442	47	171	33	0	21	11	77

✓ 1s completed at 11:38 PM