Name: Priyanka Koradkar

Task-1(Advance Level) : Develop A Neural Network That Can Read Handwriting

Importing Libraries

In [3]:
```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

In [4]:
```
(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.npz
11493376/11490434 [==============================] - 13s 1us/step
11501568/11490434 [==============================] - 13s 1us/step
```

In [5]:
```
len(X_train)
```

Out[5]:
```
60000
```

In [6]:
```
len(X_test)
```

Out[6]:
```
10000
```

In [7]:
```
X_train[0].shape
```

Out[7]:
```
(28, 28)
```

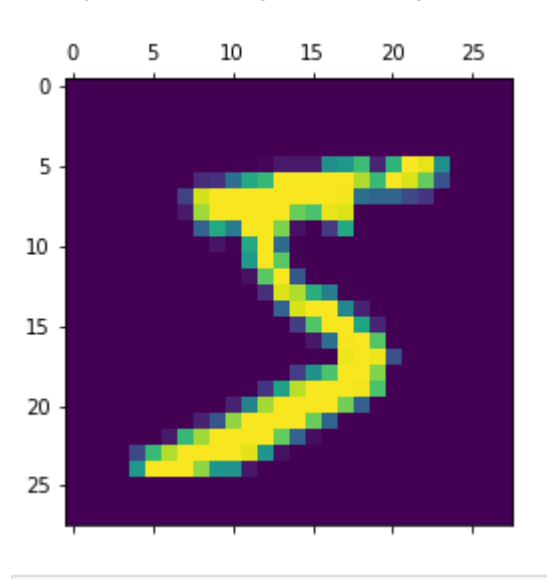
In [8]:
```
X_train
```

Out[8]:
```
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, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [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)
```

Data Visualization

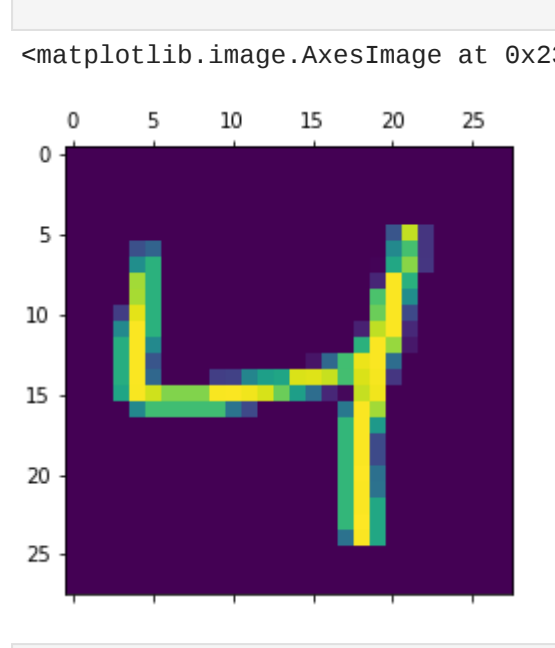
In [9]:
```
plt.matshow(X_train[0])
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x23ae19fe0d0>
```



In [10]:
```
plt.matshow(X_train[2])
```

Out[10]:
```
<matplotlib.image.AxesImage at 0x23ae3af34f0>
```



In [11]:
```
y_train[2]
```

Out[11]:
```
4
```

In [12]:
```
y_train[:6]
```

Out[12]:
```
array([5, 0, 4, 1, 9, 2], dtype=uint8)
```

Feature Scaling

In [13]:
```
# Scaling value in matrix for better accuracy
```

In [14]:
```
X_train = X_train/300
X_test = X_test/300
```

In [15]:
```
# Converted 28*28 into 1 dim i.e. flatten array
```

In [19]:
```
X_trained_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flatterned = X_test.reshape(len(X_test), 28*28)
```

In [20]:
```
X_trained_flattened.shape
```

Out[20]:
```
(60000, 784)
```

In [21]:
```
X_test_flatterned.shape
```

Out[21]:
```
(10000, 784)
```

In [22]:
```
X_trained_flattened[0]
```

Out[22]:
```
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.        , 0.01      , 0.06      , 0.06      ,
       0.06      , 0.42      , 0.45333333, 0.58333333, 0.08666667,
       0.55333333, 0.85      , 0.82333333, 0.42333333, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.1       , 0.12      , 0.31333333, 0.51333333,
       0.56666667, 0.84333333, 0.84333333, 0.84333333, 0.84333333,
       0.84333333, 0.75      , 0.57333333, 0.84333333, 0.80666667,
       0.65      , 0.21333333, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.16333333, 0.79333333,
       0.84333333, 0.84333333, 0.84333333, 0.84333333, 0.84333333,
       0.84333333, 0.84333333, 0.84333333, 0.83666667, 0.31      ,
       0.27333333, 0.27333333, 0.18666667, 0.13      , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.06      , 0.73      , 0.84333333, 0.84333333,
       0.84333333, 0.84333333, 0.84333333, 0.66      , 0.60666667,
       0.82333333, 0.80333333, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.26666667, 0.52      , 0.35666667, 0.84333333, 0.84333333,
       0.68333333, 0.03666667, 0.        , 0.14333333, 0.51333333,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.04666667,
       0.00333333, 0.51333333, 0.84333333, 0.3       , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.46333333,
       0.84333333, 0.63333333, 0.00666667, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.03666667, 0.63333333, 0.84333333,
       0.23333333, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.11666667, 0.80333333, 0.75      , 0.53333333,
       0.36      , 0.00333333, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.27      , 0.8       , 0.84333333, 0.84333333, 0.39666667,
       0.08333333, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.15      ,
       0.62      , 0.84333333, 0.84333333, 0.5       , 0.09      ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.05333333, 0.31      ,
       0.84      , 0.84333333, 0.62333333, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.83      , 0.84333333,
       0.83      , 0.21333333, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.15333333, 0.43333333,
       0.61      , 0.84333333, 0.84333333, 0.69      , 0.00666667,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.13      ,
       0.49333333, 0.76333333, 0.84333333, 0.84333333, 0.84333333,
       0.83333333, 0.60666667, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.08      , 0.38      , 0.73666667, 0.84333333, 0.84333333,
       0.84333333, 0.84333333, 0.67      , 0.26      , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.07666667, 0.22      , 0.71      , 0.84333333,
       0.84333333, 0.84333333, 0.84333333, 0.66      , 0.27      ,
       0.00666667, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.06      , 0.57      , 0.73      ,
       0.84333333, 0.84333333, 0.84333333, 0.84333333, 0.65      ,
       0.26666667, 0.03      , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.18333333, 0.57333333,
       0.75333333, 0.84333333, 0.84333333, 0.84333333, 0.84333333,
       0.81333333, 0.44333333, 0.03666667, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.45333333, 0.84333333, 0.84333333, 0.84333333,
       0.70666667, 0.45      , 0.44      , 0.05333333, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        ])
```

simple neural network

In [24]:
```
# dense is nothing but neuron in one layer connected with each neuron with other layer
model = keras.Sequential([
    keras.layers.Dense(10, input_shape=(784,), activation="sigmoid")       # 10 is output layer neuron and 784 is ip layer neuron
])

model.compile(optimizer='adam',                           # Optimizer allows to train in efficient way
            loss = "sparse_categorical_crossentropy",
            metrics = ['accuracy']
)

model.fit(X_trained_flattened, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 6s 2ms/step - loss: 0.4933 - accuracy: 0.8729
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3088 - accuracy: 0.9137
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2858 - accuracy: 0.9201
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2750 - accuracy: 0.9230
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2682 - accuracy: 0.9255
```

Out[24]:
```
<keras.callbacks.History at 0x23ae1467d90>
```

In [25]:
```
model.evaluate(X_test_flatterned, y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.2690 - accuracy: 0.9248
```

Out[25]:
```
[0.2690112888813019, 0.9247999787330627]
```

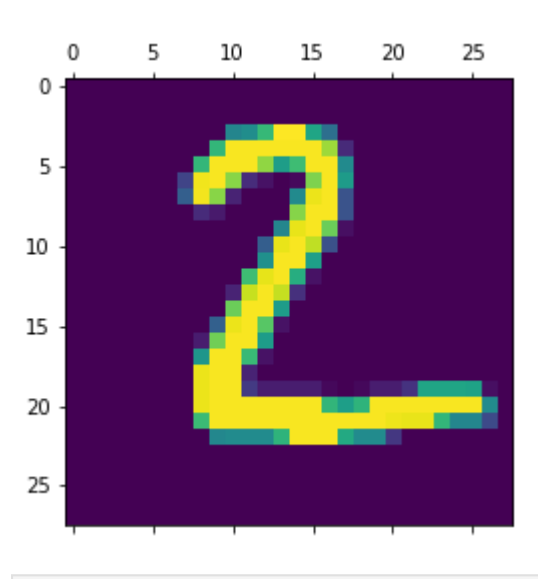
prediction

In [26]:
```
model.predict(X_test_flatterned)
```

Out[26]:
```
array([[4.4226885e-02, 8.4600754e-07, 7.4708194e-02, ..., 9.9978697e-01,
        9.7303599e-02, 5.9216118e-01],
       [5.6941450e-01, 1.1008203e-02, 9.9939460e-01, ..., 5.4270533e-12,
        2.0553422e-01, 5.3413842e-09],
       [5.4508448e-04, 9.9448633e-01, 6.7194879e-01, ..., 1.3780177e-01,
        3.8089967e-01, 4.2589724e-02],
       ...,
       [6.7041324e-06, 8.5798692e-06, 1.4815629e-03, ..., 1.7237529e-01,
        5.8300781e-01, 6.6852725e-01],
       [2.9757619e-04, 4.1314960e-04, 2.5936961e-04, ..., 5.8810627e-05,
        6.8405014e-01, 1.3580918e-04],
       [2.0065188e-02, 9.5705677e-10, 1.7541924e-01, ..., 4.5158131e-08,
        3.0782819e-04, 1.4909136e-06]], dtype=float32)
```

In [27]:
```
plt.matshow(X_test[1])
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x23ae17021f0>
```



In [28]:
```
y_pred = model.predict(X_test_flatterned)
y_pred[1]
```

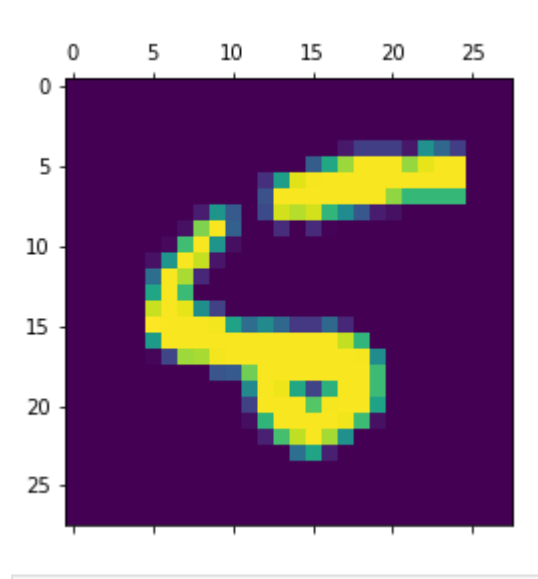
Out[28]:
```
array([5.6941450e-01, 1.1008203e-02, 9.9939460e-01, 5.8173263e-01,
       1.9634832e-09, 8.5786664e-01, 9.1079664e-01, 5.4270533e-12,
       2.0553422e-01, 5.3413842e-09], dtype=float32)
```

In [29]:
```
np.argmax(y_pred[1])
```

Out[29]:
```
2
```

In [30]:
```
# try for 8st image
plt.matshow(X_test[8])
```

Out[30]:
```
<matplotlib.image.AxesImage at 0x23a8b3e2bb0>
```



In [31]:
```
y_pred = model.predict(X_test_flatterned)
y_pred[8]
np.argmax(y_pred[8])
```

Out[31]:
```
6
```

In [33]:
```
y_predicted_labels = [np.argmax(i) for i in y_pred]
y_predicted_labels[:9]
```

Out[33]:
```
[7, 2, 1, 0, 4, 1, 4, 9, 6]
```

In [34]:
```
# confusion matrix
cm = tf.math.confusion_matrix(labels=y_test, predictions=y_predicted_labels)
```

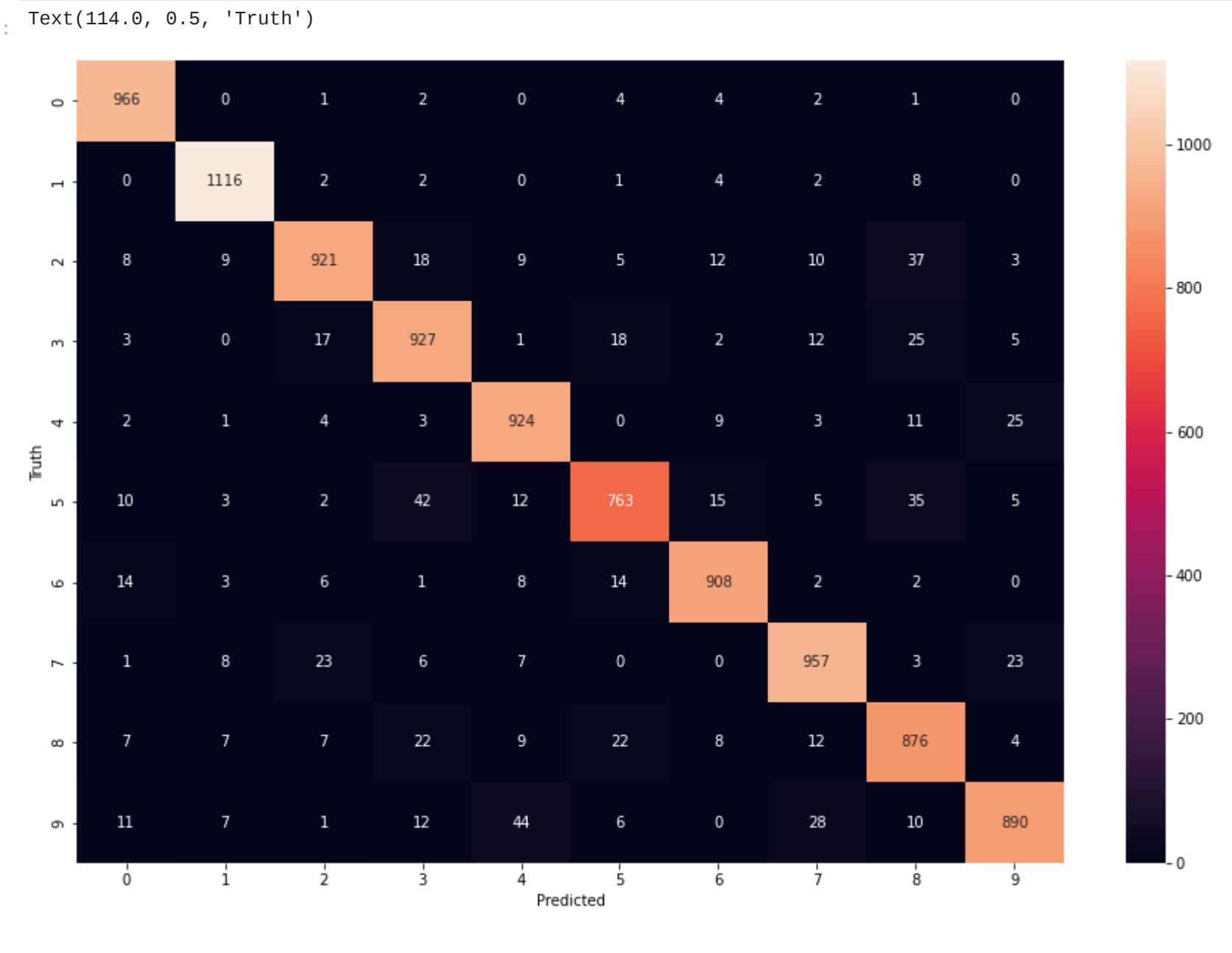
In [35]:
```
cm
```

Out[35]:
```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 966,    0,    1,    2,    0,    4,    4,    2,    1,    0],
       [   0, 1116,    2,    2,    0,    1,    4,    2,    8,    0],
       [   8,    9,  921,   18,    9,    5,   12,   10,   37,    3],
       [   3,    0,   17,  927,    1,   18,    2,   12,   25,    5],
       [   2,    1,    4,    3,  924,    0,    9,    3,   11,   25],
       [  10,    3,    2,   42,   12,  763,   15,    5,   35,    5],
       [  14,    3,    6,    1,    8,   14,  908,    2,    2,    0],
       [   1,    8,   23,    6,    7,    0,    0,  957,    3,   23],
       [   7,    7,    7,   22,    9,   22,    8,   12,  876,    4],
       [  11,    7,    1,   12,   44,    6,    0,   28,   10,  890]])>
```

visualization of confusion matrix

In [37]:
```
import seaborn as sns
plt.figure(figsize=(15,10))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel("Predicted")
plt.ylabel('Truth')
```

Out[37]:
```
Text(114.0, 0.5, 'Truth')
```



Add Hidden layer

In [38]:
```
model = keras.Sequential([
    keras.layers.Dense(100, input_shape=(784,), activation="relu"),     # Hidden layer
    keras.layers.Dense(10, activation="sigmoid")       # Output Layer
])

model.compile(optimizer='adam',                           # Optimizer allows to train in efficient way
            loss = "sparse_categorical_crossentropy",
            metrics = ['accuracy']
)

model.fit(X_trained_flattened, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2847 - accuracy: 0.9208
Epoch 2/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1291 - accuracy: 0.9617
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0900 - accuracy: 0.9727
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0698 - accuracy: 0.9793
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0546 - accuracy: 0.9835
```

Out[38]:
```
<keras.callbacks.History at 0x23a819d2e20>
```

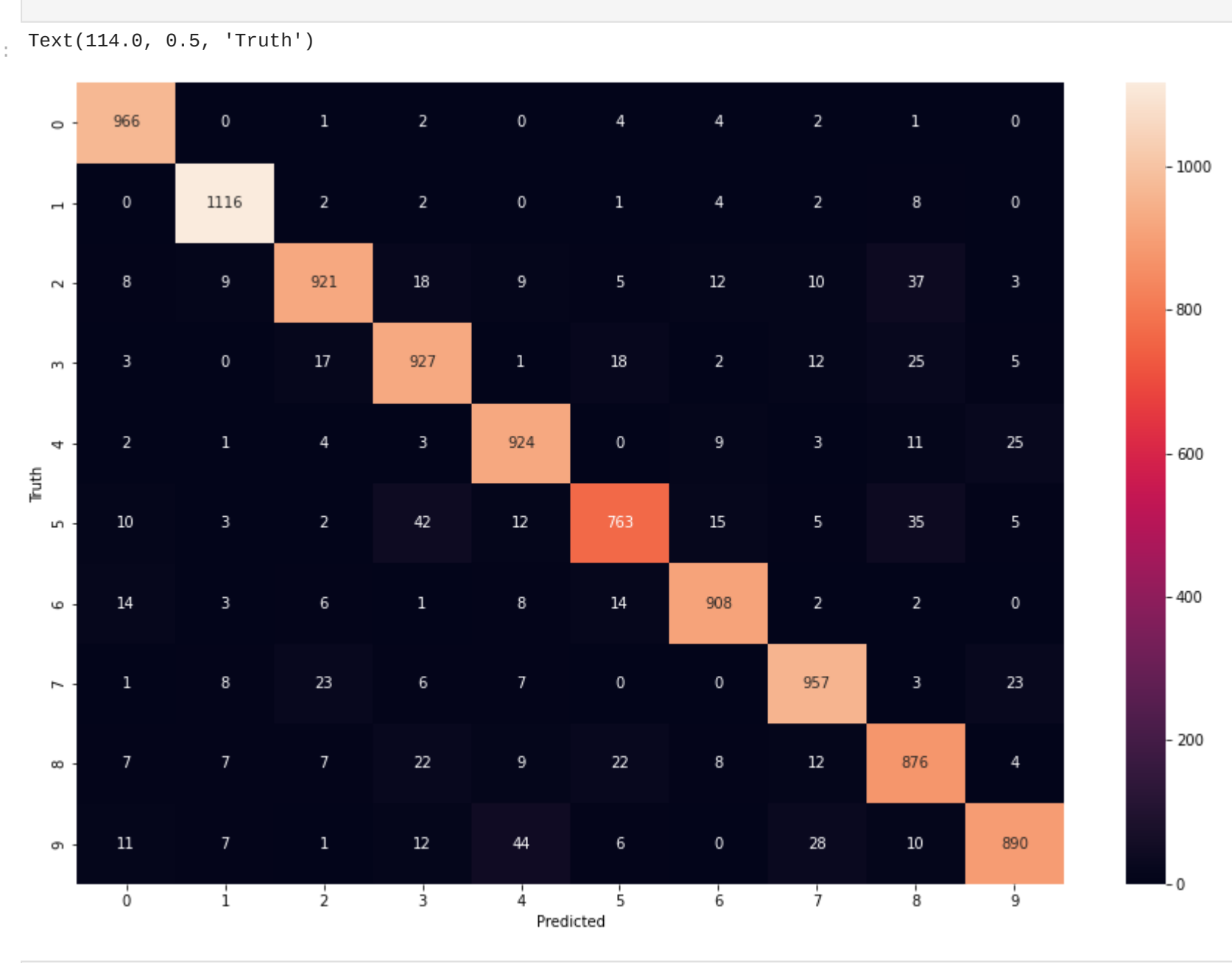
In [39]:
```
model.evaluate(X_test_flatterned, y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0772 - accuracy: 0.9746
```

Out[39]:
```
[0.07724175602197647, 0.9746000170707703]
```

In [40]:
```
# confusion matrix
y_predicted_labels = [np.argmax(i) for i in y_pred]
y_predicted_labels[:5]
cm = tf.math.confusion_matrix(labels=y_test, predictions=y_predicted_labels)

import seaborn as sns
plt.figure(figsize=(15,))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel("Predicted")
plt.ylabel('Truth')
```

Out[40]:
```
Text(114.0, 0.5, 'Truth')
```



In [ ]: