In [1]:
```
#Ayush Sharma 209303312
# Program to demonstrate back propagation algorithm in python.
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sn
```

In [2]:
```
(X_train, y_train) , (X_test, y_test) = keras.datasets.mnist.load_data()
X_train = X_train / 255
X_test = X_test / 255
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/
mnist.npz
11490434/11490434 [==============================] - 11s 1us/step
```

In [3]:
```
X_train_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flattened = X_test.reshape(len(X_test), 28*28)
X_train_flattened.shape
```

Out[3]:
```
(60000, 784)
```

In [4]:
```
model = keras.Sequential([keras.layers.Dense(10, input_shape=(784,), activation='si
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['acc
model.fit(X_train_flattened, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.4664 - accurac
y: 0.8780
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3035 - accurac
y: 0.9153
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2834 - accurac
y: 0.9211
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2731 - accurac
y: 0.9237
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2664 - accurac
y: 0.9257
```

Out[4]:
```
<keras.callbacks.History at 0x246df9ab670>
```

In [5]:
```
model.evaluate(X_test_flattened, y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.2668 - accuracy:
0.9240
```

Out[5]:
```
[0.2667868435382843, 0.9240000247955322]
```

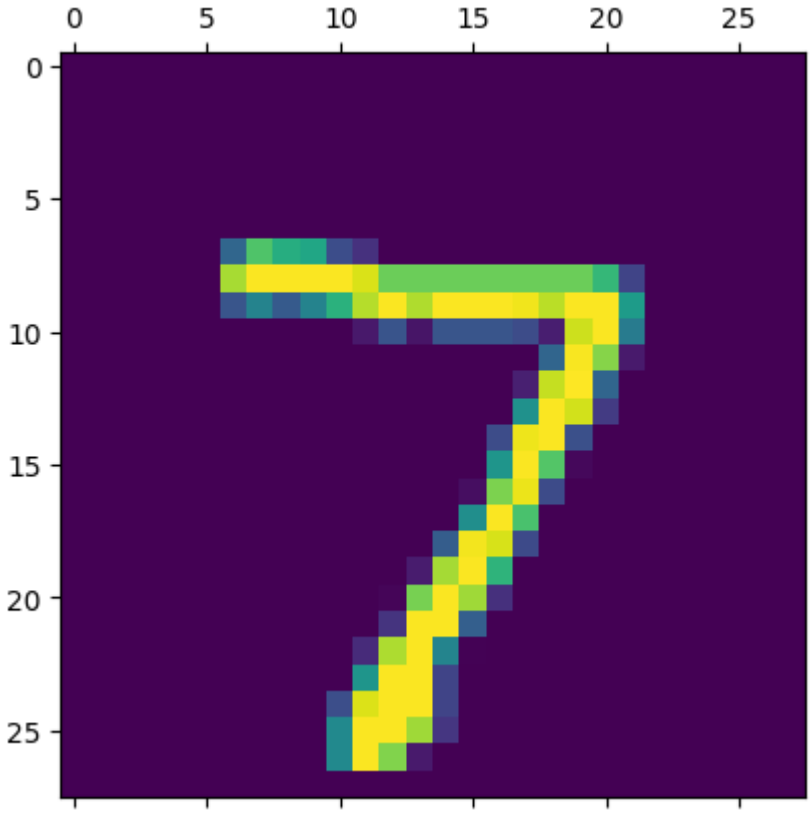
In [6]:
```
y_predicted = model.predict(X_test_flattened)
y_predicted[0]
```

```
313/313 [==============================] - 1s 2ms/step
```

Out[6]:
```
array([2.1758921e-02, 4.4378467e-07, 5.8210112e-02, 9.7385710e-01,
       3.3646403e-03, 1.3884038e-01, 1.3582170e-06, 9.9986899e-01,
       1.3372448e-01, 7.6006305e-01], dtype=float32)
```

In [7]:
```
plt.matshow(X_test[0])
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x246dfcf2f80>
```



In [8]:
```
np.argmax(y_predicted[0])
```

Out[8]:
```
7
```

In [9]:
```
y_predicted_labels = [np.argmax(i) for i in y_predicted]
y_predicted_labels[:5]
```

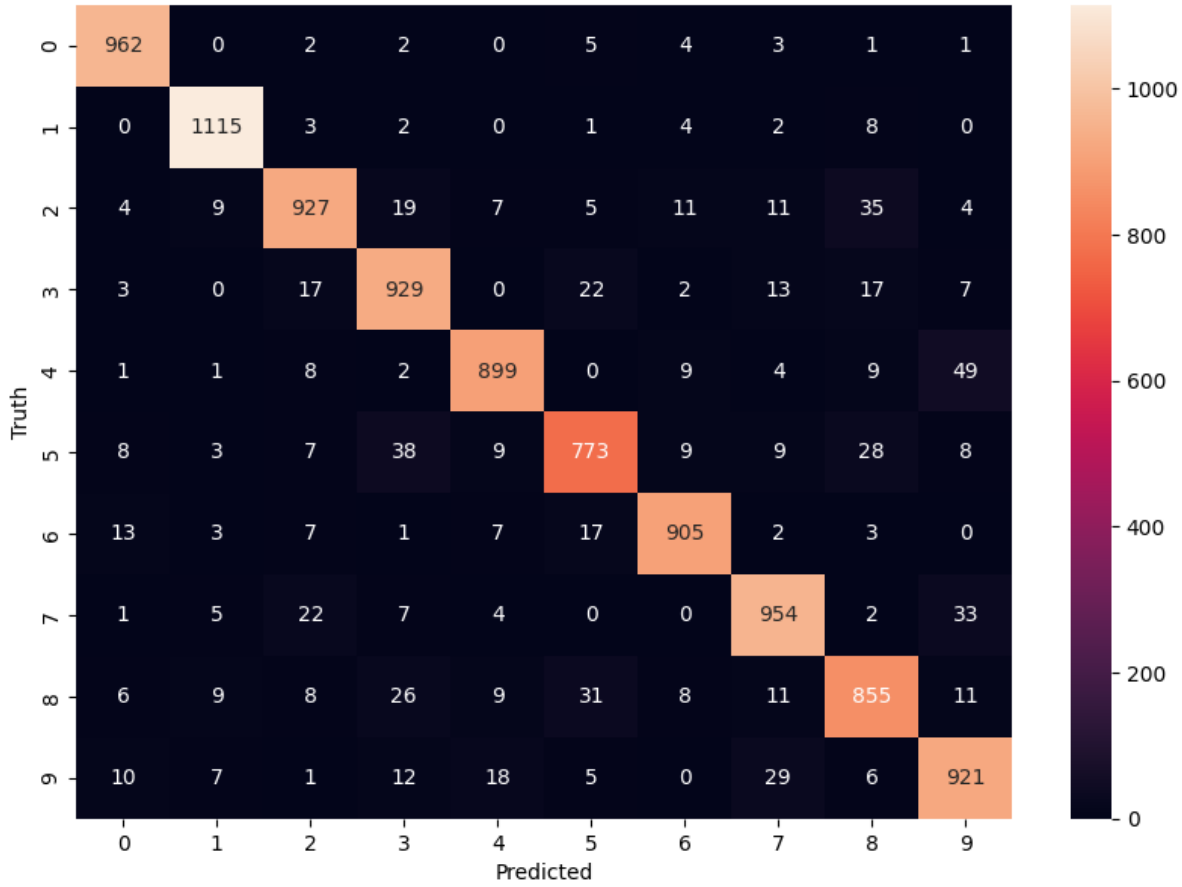
Out[9]:
```
[7, 2, 1, 0, 4]
```

In [10]:
```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
cm
```

Out[10]:
```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 962,    0,    2,    2,    0,    5,    4,    3,    1,    1],
       [   0, 1115,    3,    2,    0,    1,    4,    2,    8,    0],
       [   4,    9,  927,   19,    7,    5,   11,   11,   35,    4],
       [   3,    0,   17,  929,    0,   22,    2,   13,   17,    7],
       [   1,    1,    8,    2,  899,    0,    9,    4,    9,   49],
       [   8,    3,    7,   38,    9,  773,    9,    9,   28,    8],
       [  13,    3,    7,    1,    7,   17,  905,    2,    3,    0],
       [   1,    5,   22,    7,    4,    0,    0,  954,    2,   33],
       [   6,    9,    8,   26,    9,   31,    8,   11,  855,   11],
       [  10,    7,    1,   12,   18,    5,    0,   29,    6,  921]])>
```

In [11]:
```
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[11]:
```
Text(95.72222222222221, 0.5, 'Truth')
```



In [12]:
```
model = keras.Sequential([keras.layers.Dense(100, input_shape=(784,), activation='r
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['ac
model.fit(X_train_flattened, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2736 - accurac
y: 0.9226
Epoch 2/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1246 - accurac
y: 0.9639
Epoch 3/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0878 - accurac
y: 0.9743
Epoch 4/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0662 - accurac
y: 0.9800
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0525 - accurac
y: 0.9838
```

Out[12]:
```
<keras.callbacks.History at 0x246dfc1bf70>
```

In [13]:
```
model.evaluate(X_test_flattened,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0732 - accuracy:
0.9773
```

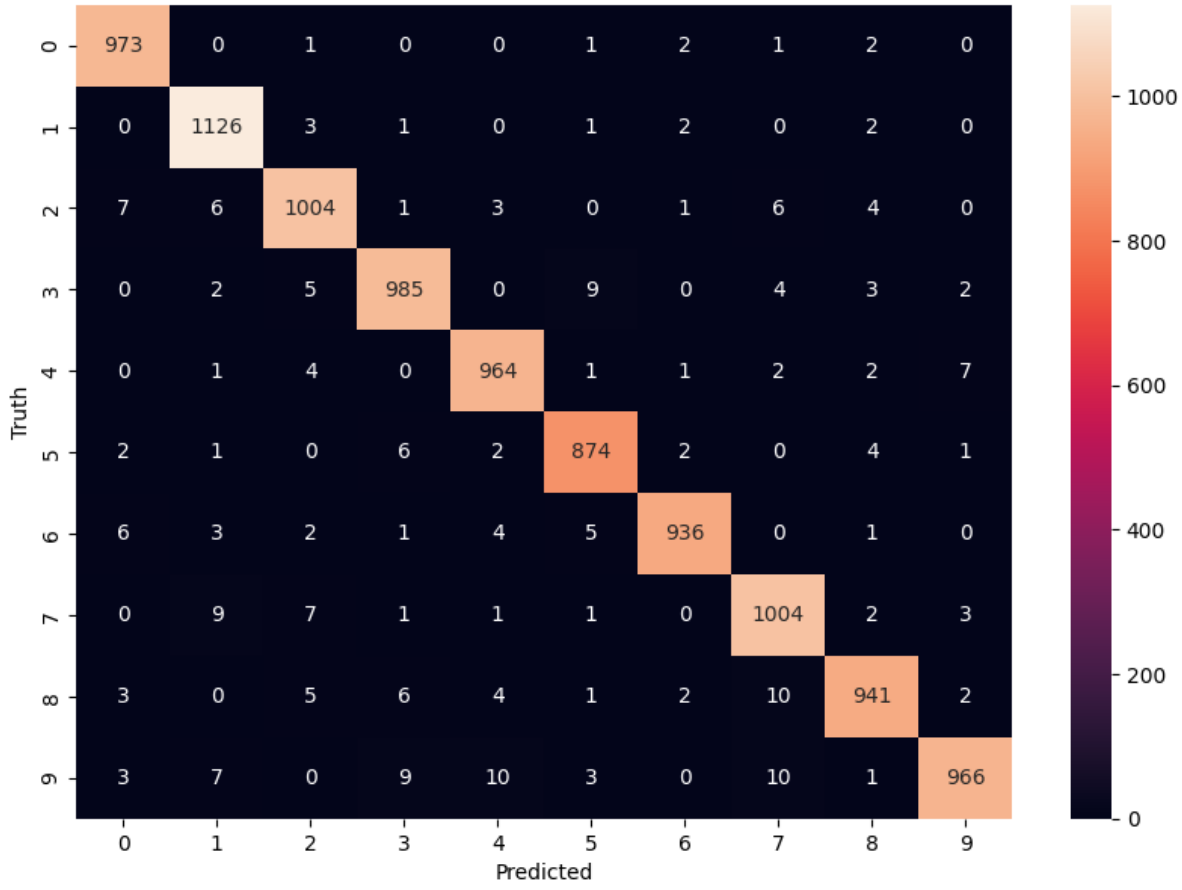
Out[13]:
```
[0.07323426008224487, 0.9772999882698059]
```

In [14]:
```
y_predicted = model.predict(X_test_flattened)
y_predicted_labels = [np.argmax(i) for i in y_predicted]
```

```
313/313 [==============================] - 1s 2ms/step
```

In [15]:
```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[15]:
```
Text(95.72222222222221, 0.5, 'Truth')
```



In [17]:
```
model = keras.Sequential([keras.layers.Flatten(input_shape=(28, 28)),keras.layers.D
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['acc
model.fit(X_train, y_train, epochs=10)
model.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2714 - accurac
y: 0.9231
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1238 - accurac
y: 0.9634
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0854 - accurac
y: 0.9738
Epoch 4/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0664 - accurac
y: 0.9790
Epoch 5/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0525 - accurac
y: 0.9837
Epoch 6/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0408 - accurac
y: 0.9872
Epoch 7/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0348 - accurac
y: 0.9895
Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0267 - accurac
y: 0.9919
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0226 - accurac
y: 0.9935
Epoch 10/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0200 - accurac
y: 0.9936
313/313 [==============================] - 1s 2ms/step - loss: 0.0905 - accuracy:
0.9763
```

Out[17]: `[0.09054549783468246, 0.9763000011444092]`