```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
len(x_train)
```

```
60000
```

```
len(x_test)
```

```
10000
```

```
x_train.shape
```

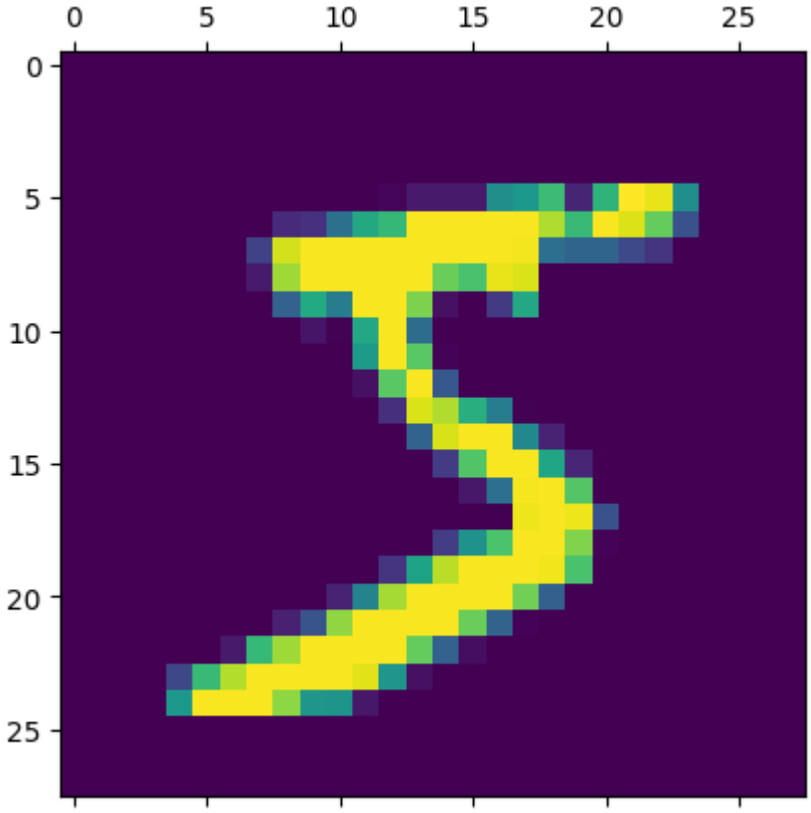
```
(60000, 28, 28)
```

```
x_train[0]
```

```
ndarray (28, 28) show data
```

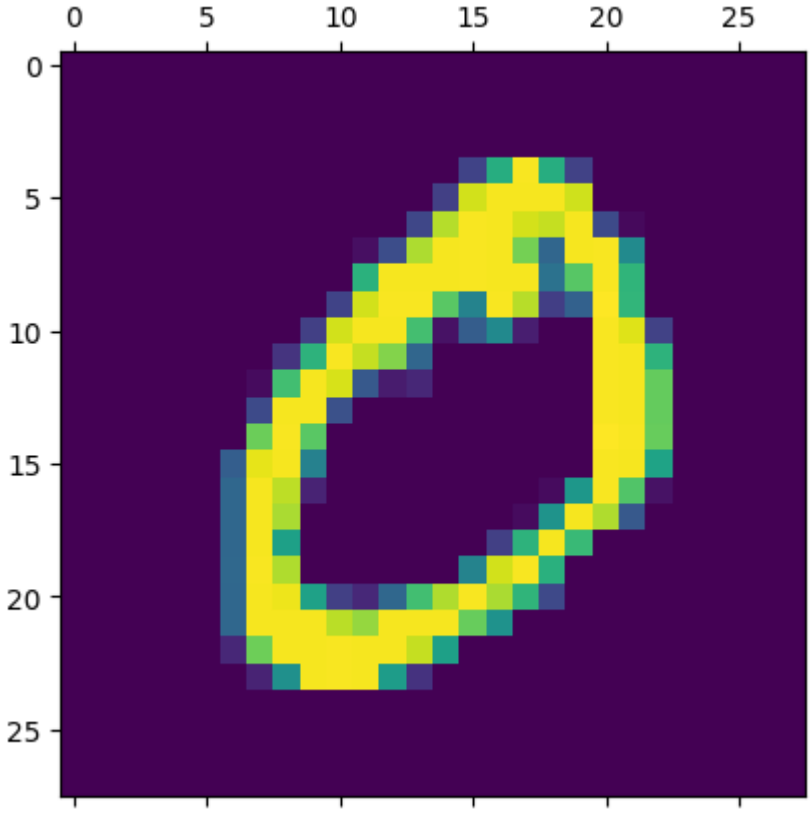
```
plt.matshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7c4c8ac07190>
```



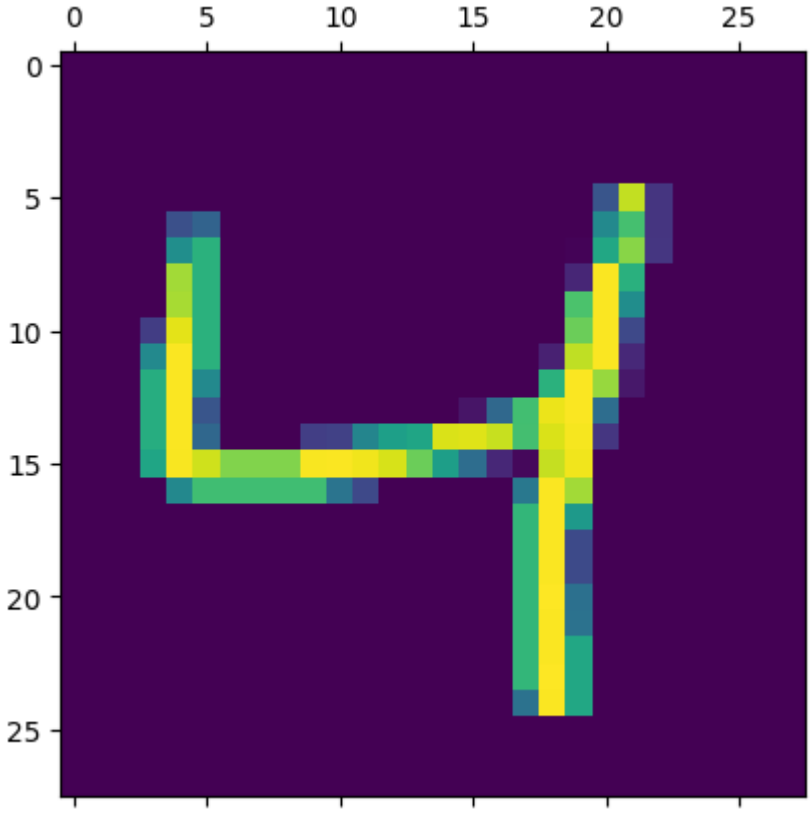
```
plt.matshow(x_train[1])
```

```
<matplotlib.image.AxesImage at 0x7c4c8aca6f90>
```



```
plt.matshow(x_train[2])
```

```
<matplotlib.image.AxesImage at 0x7c4c8ac9add0>
```




```
y_train[2]
```

```
np.uint8(4)
```

```
y_train[:5]
```

```
array([5, 0, 4, 1, 9], dtype=uint8)
```

```
x_train.shape
```

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

scalling the data

```
x_train = x_train / 255
x_test = x_test / 255
```

```
x_train_flattened = x_train.reshape(60000, 28 * 28)
x_test_flattened = x_test.reshape(10000, 28 * 28)
```

```
x_train_flattened.shape
```

```
(60000, 784)
```

```
x_test_flattened.shape
```

```
(10000, 784)
```

```
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_flattened, y_train, epochs=5)
```

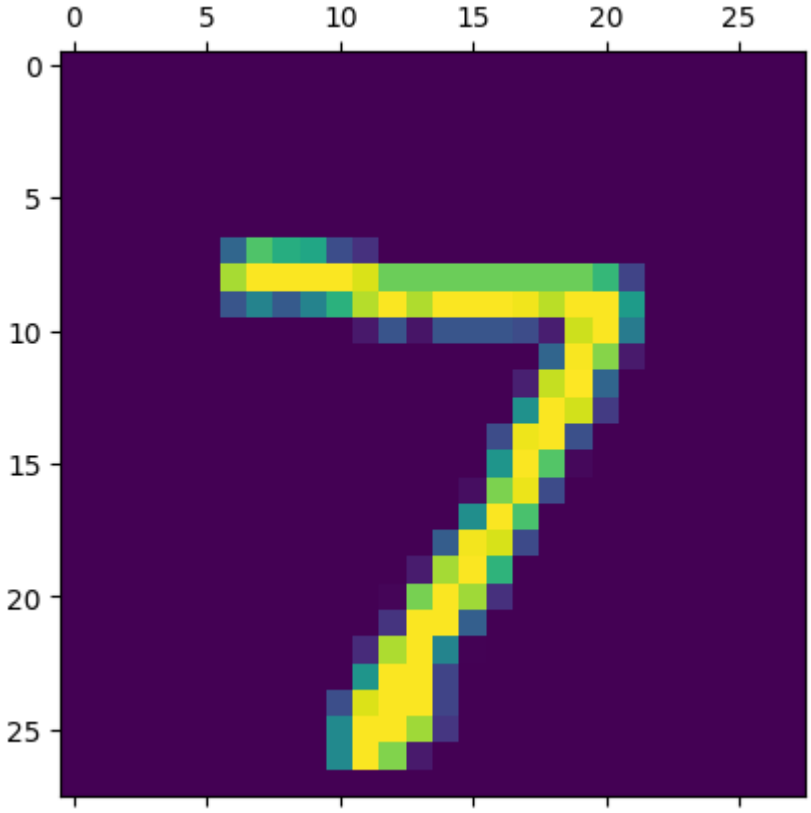
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:93: Use
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8158 - loss: 0.7228
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9136 - loss: 0.3107
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9187 - loss: 0.2884
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9227 - loss: 0.2747
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9254 - loss: 0.2654
<keras.src.callbacks.history.History at 0x7c4c9093b950>
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9126 - loss: 0.3056
[0.2687021791934967, 0.925000011920929]
```

```
plt.matshow(x_test[0])
```

```
<matplotlib.image.AxesImage at 0x7c4c85b0dc50>
```




```
y_predicted = model.predict(x_test_flattened)
y_predicted[0]
```

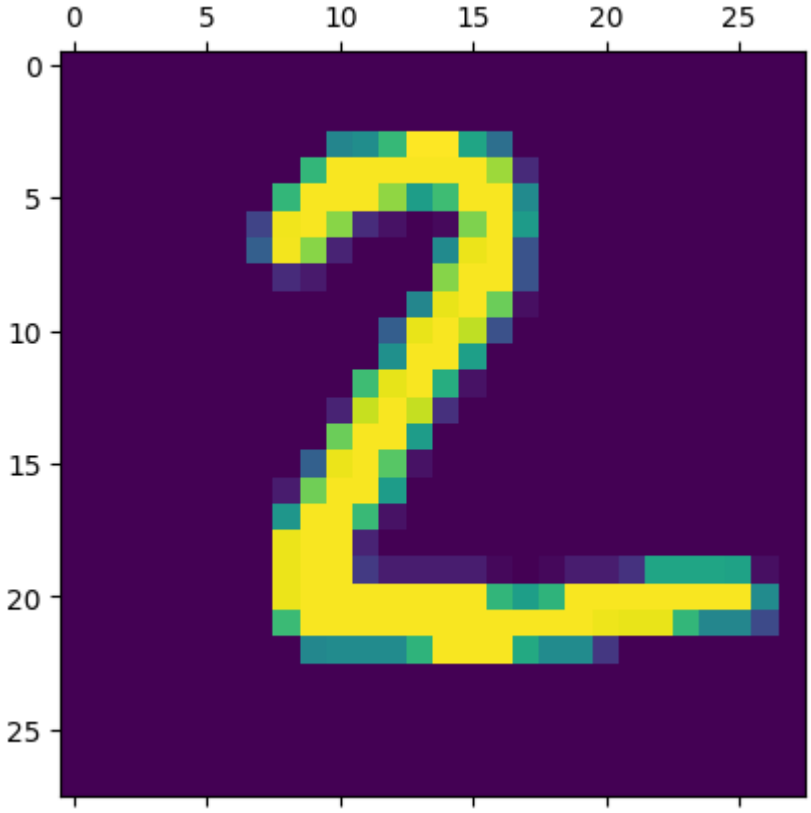
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
array([3.7691776e-02, 3.6071768e-07, 6.0681131e-02, 9.6168643e-01,
       1.6371487e-03, 8.0357373e-02, 2.2784391e-06, 9.9973071e-01,
       8.6109348e-02, 5.7833678e-01], dtype=float32)
```

```
np.argmax(y_predicted[0])
```

```
np.int64(7)
```

```
plt.matshow(x_test[1])
```

```
<matplotlib.image.AxesImage at 0x7c4c8569d610>
```




```
y_predicted = model.predict(x_test_flattened)
y_predicted[1]
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
array([4.0558359e-01, 4.4072373e-03, 9.9965429e-01, 3.3885807e-01,
       9.4369013e-10, 7.4571568e-01, 8.3298868e-01, 6.4183197e-13,
       1.3936487e-01, 2.0717528e-09], dtype=float32)
```

```
np.argmax(y_predicted[1])
```

```
np.int64(2)
```

```
y_predicted_lbels = [np.argmax(i) for i in y_predicted]
y_predicted_lbels[:5]
```

```
[np.int64(7), np.int64(2), np.int64(1), np.int64(0), np.int64(4)]
```

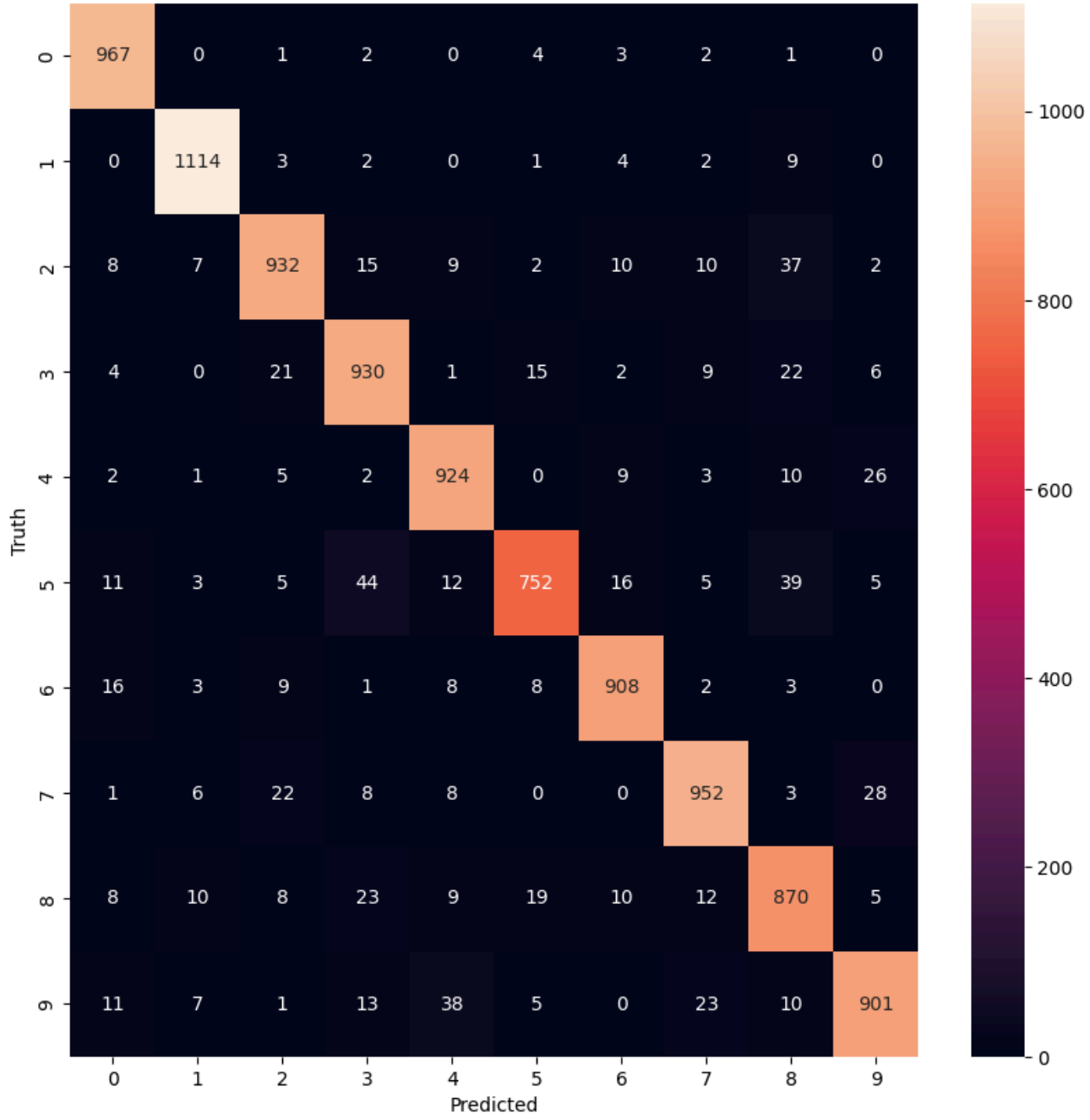
```
cm = tf.math.confusion_matrix(labels = y_test, predictions = y_predicted_lbels)
cm
```

```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 967,    0,    1,    2,    0,    4,    3,    2,    1,    0],
       [   0, 1114,    3,    2,    0,    1,    4,    2,    9,    0],
       [   8,    7,  932,   15,    9,    2,   10,   10,   37,    2],
       [   4,    0,   21,  930,    1,   15,    2,    9,   22,    6],
       [   2,    1,    5,    2,  924,    0,    9,    3,   10,   26],
       [  11,    3,    5,   44,   12,  752,   16,    5,   39,    5],
       [  16,    3,    9,    1,    8,    8,  908,    2,    3,    0],
       [   1,    6,   22,    8,    8,    0,    0,  952,    3,   28],
```

```
             [   8,   10,    8,   23,    9,   19,   10,   12,  870,    5],
             [  11,    7,    1,   13,   38,    5,    0,   23,   10,  901]],
            dtype=int32)>
```

```
import seaborn as sn
plt.figure(figsize = (10,10))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(95.72222222222221, 0.5, 'Truth')
```



adding a 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_flattened, y_train, epochs=5)
```

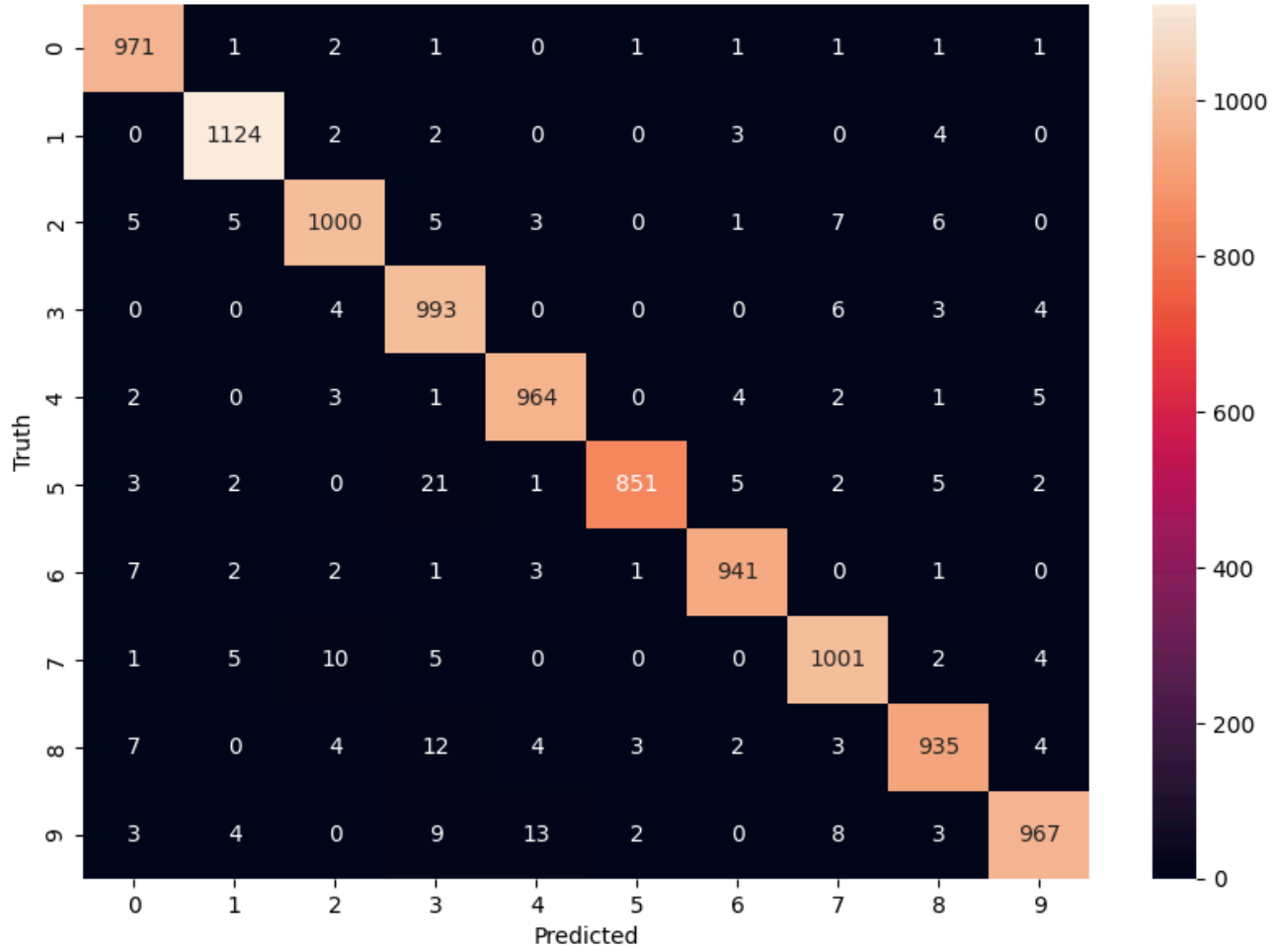
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:93: Use
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8716 - loss: 0.4453
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9614 - loss: 0.1309
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9729 - loss: 0.0894
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9798 - loss: 0.0671
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9851 - loss: 0.0483
<keras.src.callbacks.history.History at 0x7c4c8aca6150>
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9699 - loss: 0.0967
[0.08165591210126877, 0.9746999740600586]
```

```
y_predicted = model.predict(x_test_flattened)
y_predicted_lbels = [np.argmax(i) for i in y_predicted]
cm = tf.math.confusion_matrix(labels = y_test, predictions = y_predicted_lbels)
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Text(95.72222222222221, 0.5, 'Truth')
```



to avoid flattern arraay

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(100, activation='relu'),#hidden layer
    keras.layers.Dense(10, activation='sigmoid')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py
  super().__init__(**kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8756 - loss: 0.4463
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9641 - loss: 0.1265
Epoch 3/5
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9761 - loss: 0.0828
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9816 - loss: 0.0608
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9868 - loss: 0.0457
<keras.src.callbacks.history.History at 0x7c4c6546ef50>
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.