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

```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
len(X_train)
```

```
60000
```

```
len(X_test)
```

```
10000
```

```
X_train[0].shape
```

```
(28, 28)
```

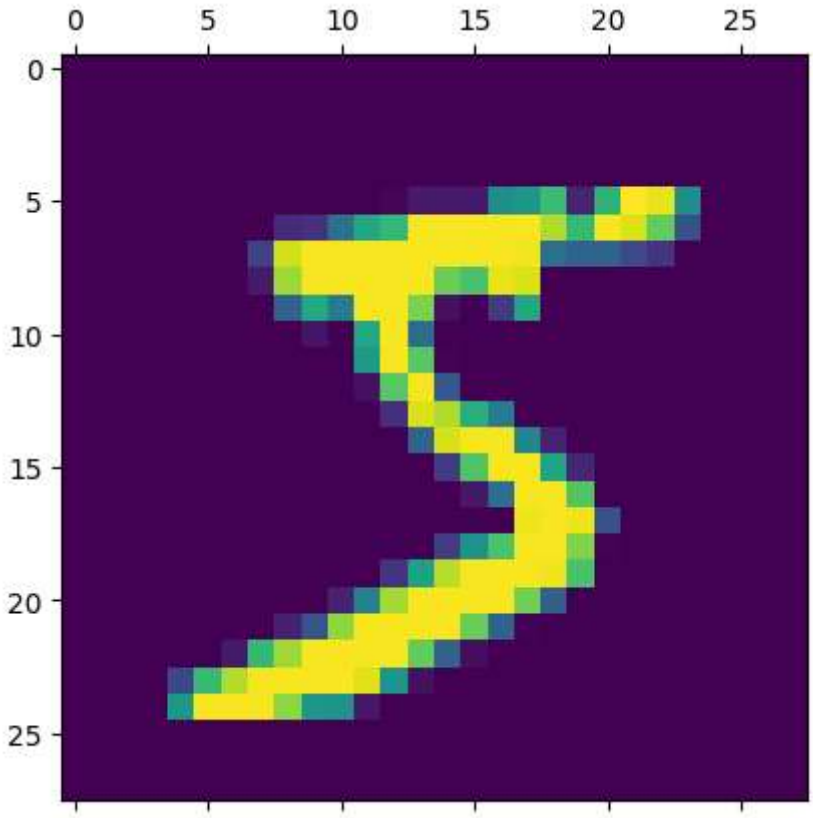
```
X_train[0]
```

```
ndarray (28, 28) hide data
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)
```

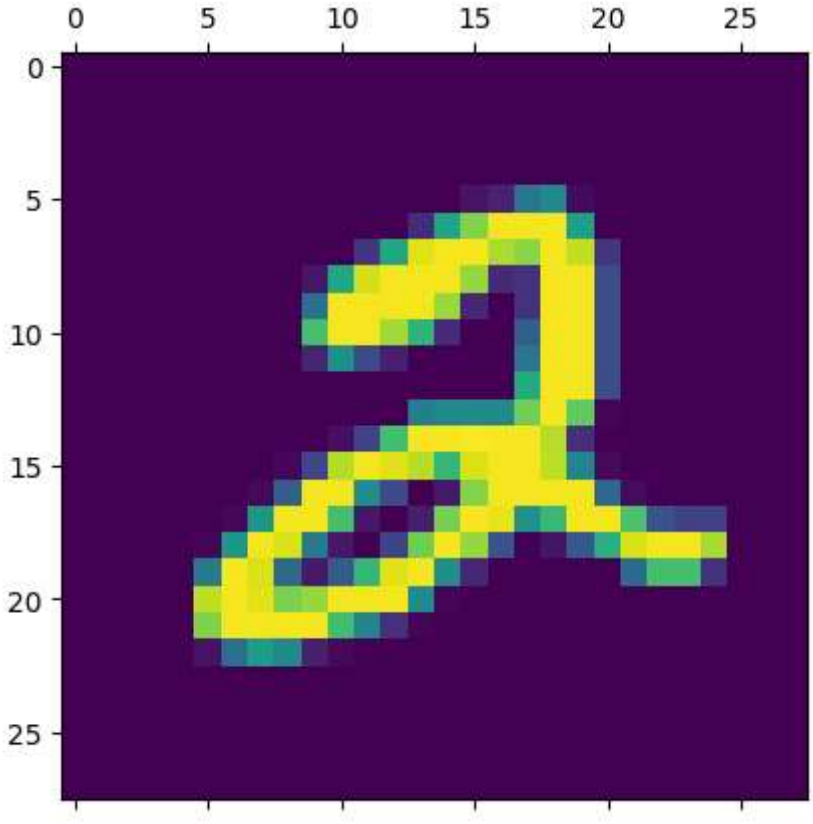
```
plt.matshow(X_train[0])
```

```
<matplotlib.image.AxesImage at 0x78153d2d09a0>
```




```
plt.matshow(X_train[5])
```

```
<matplotlib.image.AxesImage at 0x78153d57bcd0>
```




```
y_train[2]
```

```
4
```

```
y_train[:5]
```

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

```
X_train = X_train / 255
X_test = X_test / 255
```

```
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
```

```
X_train[0].shape
```

```
(784,)
```

```
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten,Dense
```

```
model  = Sequential()

model.add(Dense(10, activation='softmax',input_shape=(784,)))
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(X_train,y_train,epochs=5)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8072 - loss: 0.7267
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9151 - loss: 0.3079
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9199 - loss: 0.2894
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9236 - loss: 0.2722
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9268 - loss: 0.2650
<keras.src.callbacks.history.History at 0x78153d464eb0>
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9143 - loss: 0.3023
[0.26665347814559937, 0.9254000186920166]
```

```
model.predict(X_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
array([[5.6211493e-06, 7.9625834e-11, 2.0384223e-05, ..., 9.9146736e-01,
        3.1063890e-05, 4.6331936e-04],
       [2.3813271e-04, 2.6575958e-06, 9.8944682e-01, ..., 5.6023453e-16,
        1.2607417e-04, 1.1237867e-12],
       [2.2342490e-06, 9.7659731e-01, 1.0822067e-02, ..., 9.6586457e-04,
        4.8037283e-03, 3.8969322e-04],
       ...,
       [1.1733340e-08, 1.7358657e-08, 6.3683083e-06, ..., 1.2374662e-03,
        7.1569551e-03, 2.4908291e-02],
       [4.9370118e-07, 5.9014070e-07, 9.2638209e-07, ..., 1.7624758e-07,
        1.1376546e-02, 1.2455305e-06],
       [1.1454872e-06, 5.3599926e-14, 4.9303831e-05, ..., 3.4453621e-12,
        3.4974370e-08, 1.5631410e-10]], dtype=float32)
```

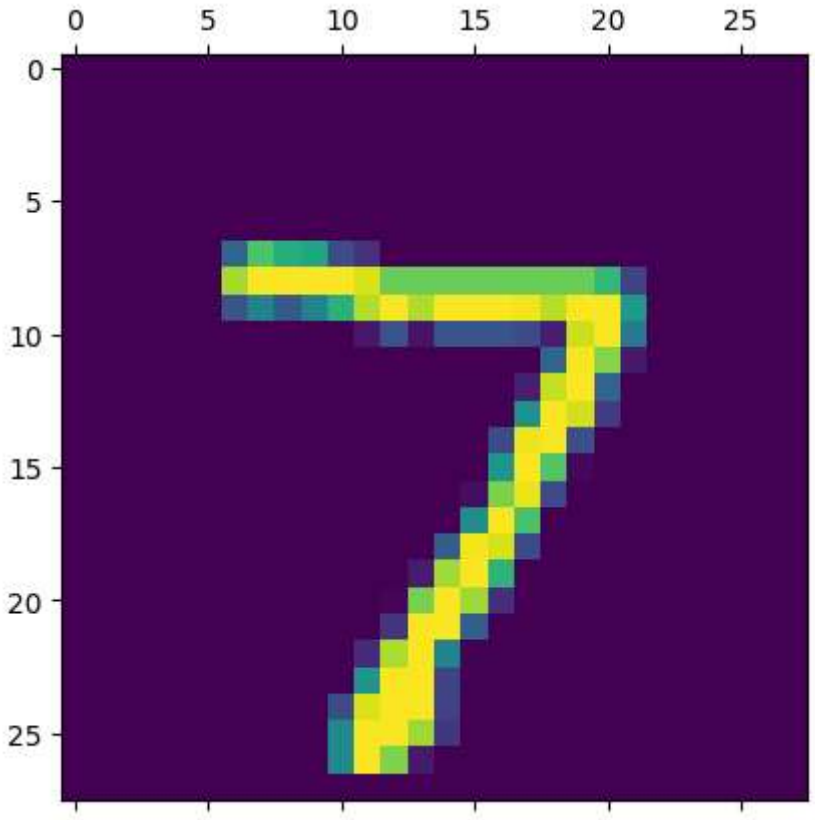
```
plt.matshow(X_test[0])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-26-3fb0095ace10> in <cell line: 1>()
----> 1 plt.matshow(X_test[0])

                                 1 frames
/usr/local/lib/python3.10/dist-packages/matplotlib/figure.py in figaspect(arg)
   3604     # Extract the aspect ratio of the array
   3605     if isarray:
-> 3606         nr, nc = arg.shape[:2]
   3607         arr_ratio = nr / nc
   3608     else:

ValueError: not enough values to unpack (expected 2, got 1)
```

Next steps: Explain error

```
import matplotlib.pyplot as plt
# Reshape X_test[0] to a 2D array before plotting.
# Assuming X_test[0] represents a 28x28 image, reshape it accordingly.
plt.matshow(X_test[0].reshape(28, 28))
```

```
<matplotlib.image.AxesImage at 0x78153d92ec50>
```




```
y_predicted = model.predict(X_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```

```
y_predicted[0]
```

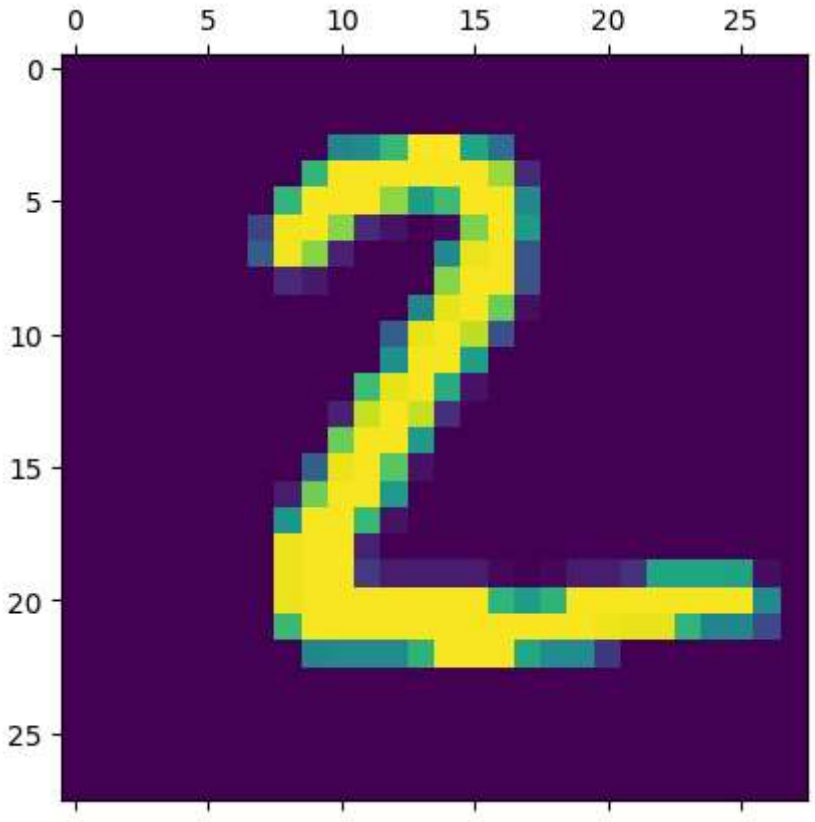
```
array([5.6211493e-06, 7.9625834e-11, 2.0384223e-05, 7.9663293e-03,
       6.1376062e-07, 4.5238870e-05, 5.0766430e-10, 9.9146736e-01,
       3.1063890e-05, 4.6331936e-04], dtype=float32)
```

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

```
7
```

```
plt.matshow(X_test[1].reshape(28, 28))
```

```
<matplotlib.image.AxesImage at 0x7815415d4280>
```




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

```
2
```

```
tf.math.confusion_matrix(y_test,np.argmax(y_predicted))
```

```
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
<ipython-input-33-e807d09a9d81> in <cell line: 1>()
----> 1 tf.math.confusion_matrix(y_test,np.argmax(y_predicted))

                                 1 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/python/framework/ops.py in
raise_from_not_ok_status(e, name)
   5981 def raise_from_not_ok_status(e, name) -> NoReturn:
   5982   e.message += (" name: " + str(name if name is not None else ""))
-> 5983   raise core._status_to_exception(e) from None  # pylint: disable=protected-
access
   5984
   5985

InvalidArgumentError: {{function_node
__wrapped__Pack_N_2_device_/job:localhost/replica:0/task:0/device:CPU:0}} Shapes of all
inputs must match: values[0].shape = [10000] != values[1].shape = [] [Op:Pack] name:
stack
```

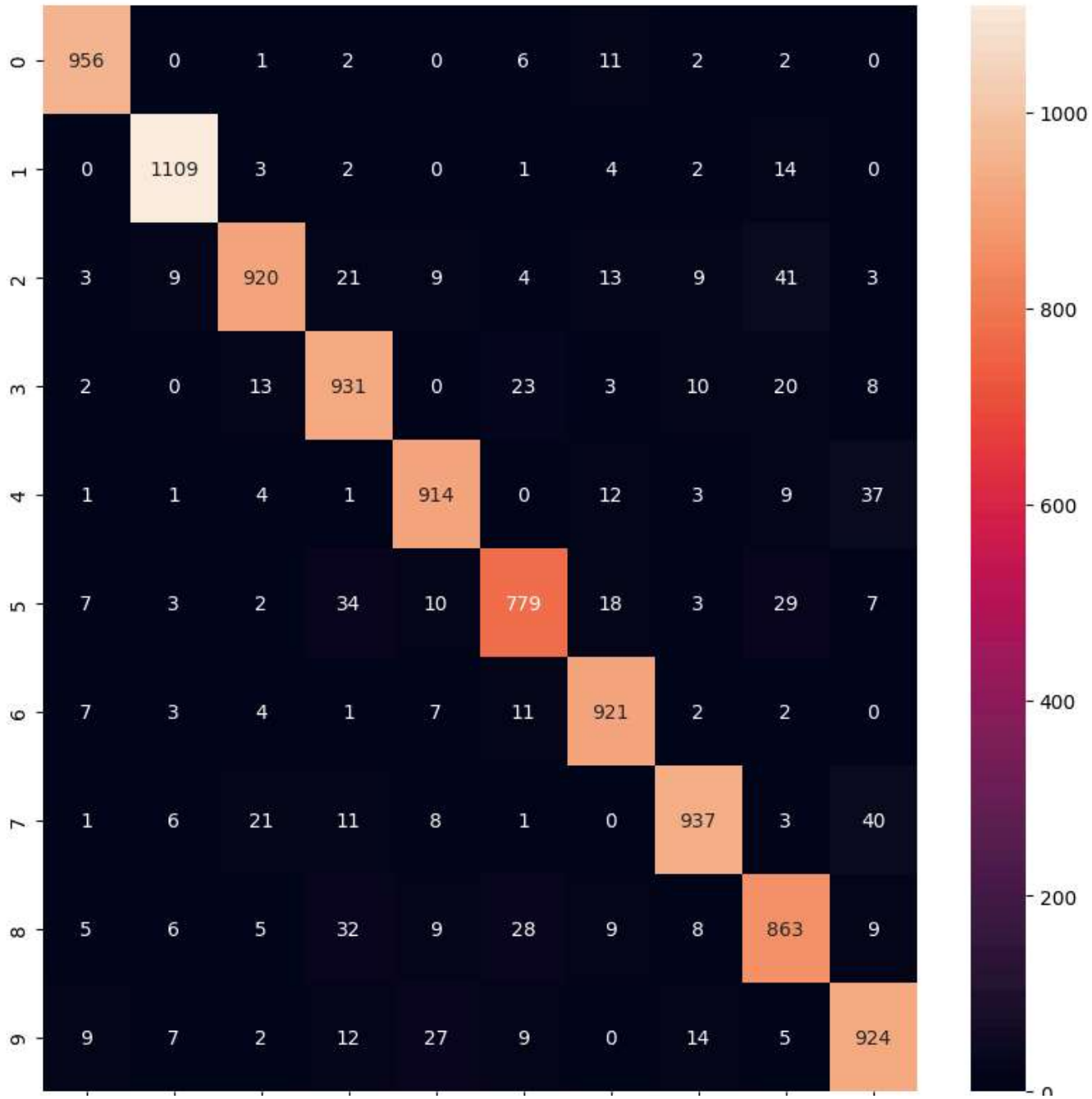
Next steps: Explain error

```
tf.math.confusion_matrix(y_test, np.argmax(y_predicted, axis=1))
```

```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 956,    0,    1,    2,    0,    6,   11,    2,    2,    0],
       [   0, 1109,    3,    2,    0,    1,    4,    2,   14,    0],
       [   3,    9,  920,   21,    9,    4,   13,    9,   41,    3],
       [   2,    0,   13,  931,    0,   23,    3,   10,   20,    8],
       [   1,    1,    4,    1,  914,    0,   12,    3,    9,   37],
       [   7,    3,    2,   34,   10,  779,   18,    3,   29,    7],
       [   7,    3,    4,    1,    7,   11,  921,    2,    2,    0],
       [   1,    6,   21,   11,    8,    1,    0,  937,    3,   40],
       [   5,    6,    5,   32,    9,   28,    9,    8,  863,    9],
       [   9,    7,    2,   12,   27,    9,    0,   14,    5,  924]],
      dtype=int32)>
```

```
import seaborn as sn
plt.figure(figsize = (10,10))
sn.heatmap(tf.math.confusion_matrix(y_test, np.argmax(y_predicted, axis=1)), annot=True, fmt
plt
```

<module 'matplotlib.pyplot' from '/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py'>




```
model  = Sequential()

model.add(Dense(100, activation='relu',input_shape=(784,)))
model.add(Dense(10,activation='relu'))
model.add(Dense(10, activation='sigmoid'))
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(X_train,y_train,epochs=10)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7852 - loss: 0.6714
Epoch 2/10
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9606 - loss: 0.1334
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9714 - loss: 0.0936
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9785 - loss: 0.0690
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9820 - loss: 0.0591
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9867 - loss: 0.0443
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9878 - loss: 0.0372
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9898 - loss: 0.0323
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9921 - loss: 0.0247
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9934 - loss: 0.0205
<keras.src.callbacks.history.History at 0x781524ff2980>
```

Start coding or generate with AI.

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9708 - loss: 0.1061
[0.09192464500665665, 0.9742000102996826]
```

```
y_predicted = model.predict(X_test)
tf.math.confusion_matrix(y_test, np.argmax(y_predicted, axis=1))

plt.figure(figsize = (10,10))
sn.heatmap(tf.math.confusion_matrix(y_test, np.argmax(y_predicted, axis=1)), annot
plt
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.10/dist-
packages/matplotlib/pyplot.py'>
```