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

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

```
len(x_train)
```

```
60000
```

```
len(x_test)
```

```
10000
```

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

```
(28, 28)
```

```
x_train[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,
3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,
0,
          0,   0],
```

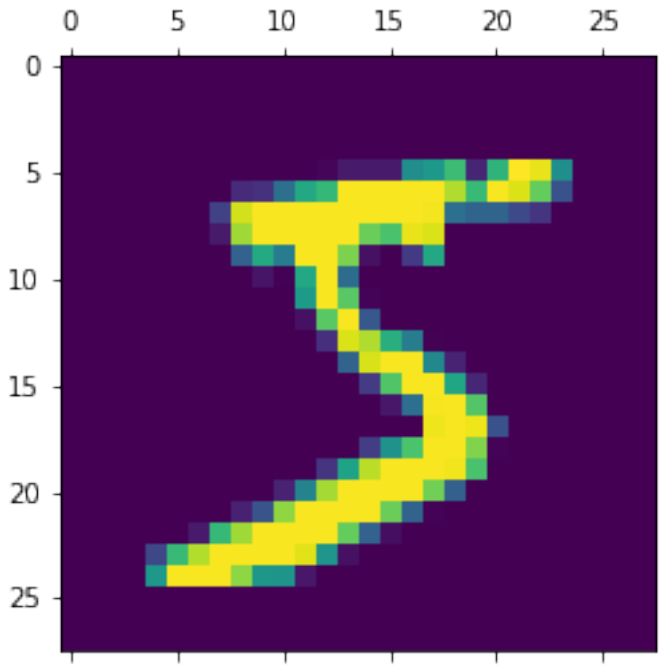
```
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154,
170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253,
253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   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 0x22c2cd0be20>
```




```
y_train[0]
```

```
5
```

```
y_train[:5]
```

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

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

```
x_train_flat = x_train.reshape(len(x_train),28*28)
```

```
x_train_flat
```

```
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.]])
```

```
x_test_flat = x_test.reshape(len(x_test),28*28)
```

```
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_flat,y_train,epochs=5)
```

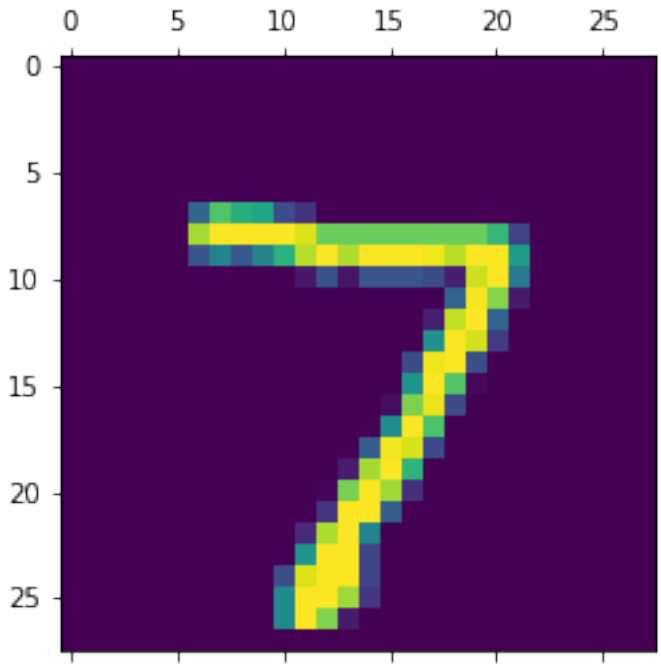
```
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss:
0.4609 - accuracy: 0.8795
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.3036 - accuracy: 0.9151
Epoch 3/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.2834 - accuracy: 0.9206
Epoch 4/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.2730 - accuracy: 0.9237
Epoch 5/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.2666 - accuracy: 0.9259

<keras.callbacks.History at 0x22c2c78fb80>
```

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

```
313/313 [==============================] - 1s 1ms/step - loss: 0.2728
- accuracy: 0.9217

[0.27284231781959534, 0.9217000007629395]
```

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

```
<matplotlib.image.AxesImage at 0x22c2cc3f0d0>
```




```
y_pred = model.predict(x_test_flat)
```

```
313/313 [==============================] - 0s 1ms/step
```

```
y_pred[0]
```

```
array([2.3314869e-02, 2.4301428e-07, 5.7534065e-02, 9.1192228e-01,
       2.6460686e-03, 1.3798307e-01, 1.2737116e-06, 9.9980021e-01,
       8.8699818e-02, 6.5576893e-01], dtype=float32)
```

```
answer = np.argmax(y_pred[0])
answer
```

```
7
```

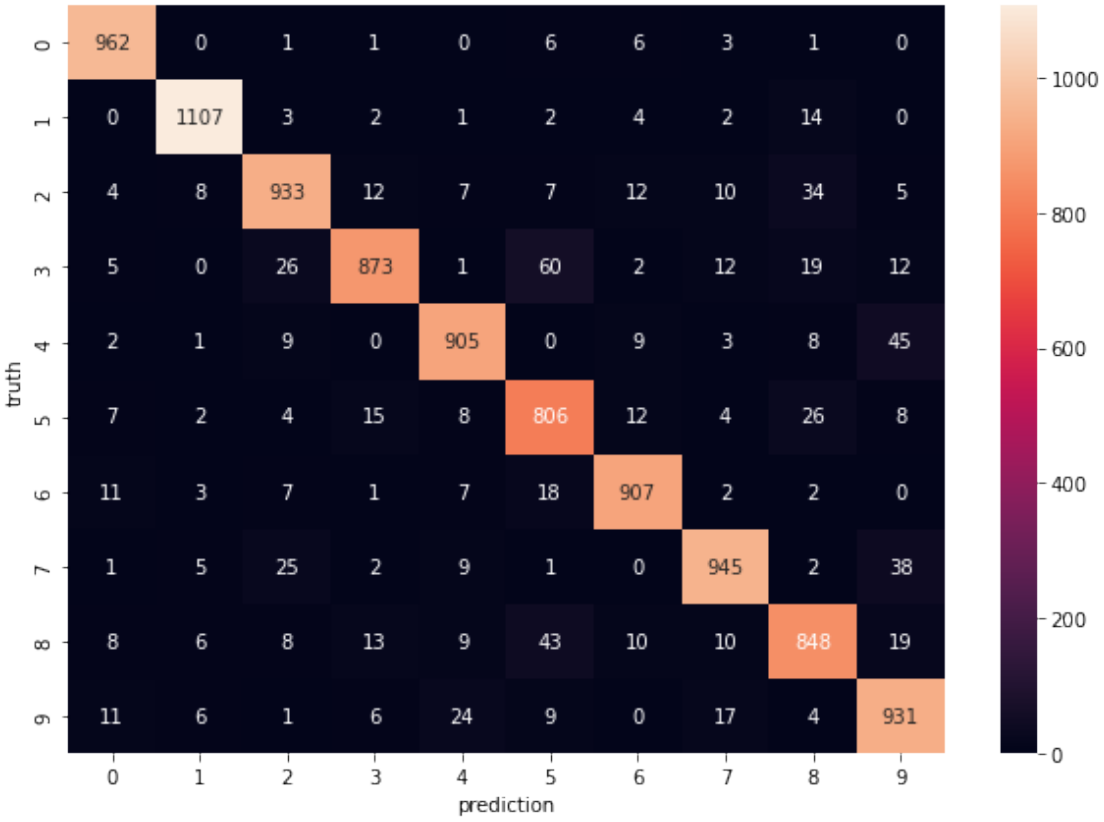
```
y_pred_labels = []
for i in range(len(y_pred)):
    y_pred_labels.append(np.argmax(y_pred[i]))
y_pred_labels[0]
```

```
7
```

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

```
import seaborn as sn
plt.figure(figsize= (10,7))
sn.heatmap(cm,annot=True,fmt='d')
plt.xlabel('prediction')
plt.ylabel('truth')
```

```
Text(69.0, 0.5, 'truth')
```

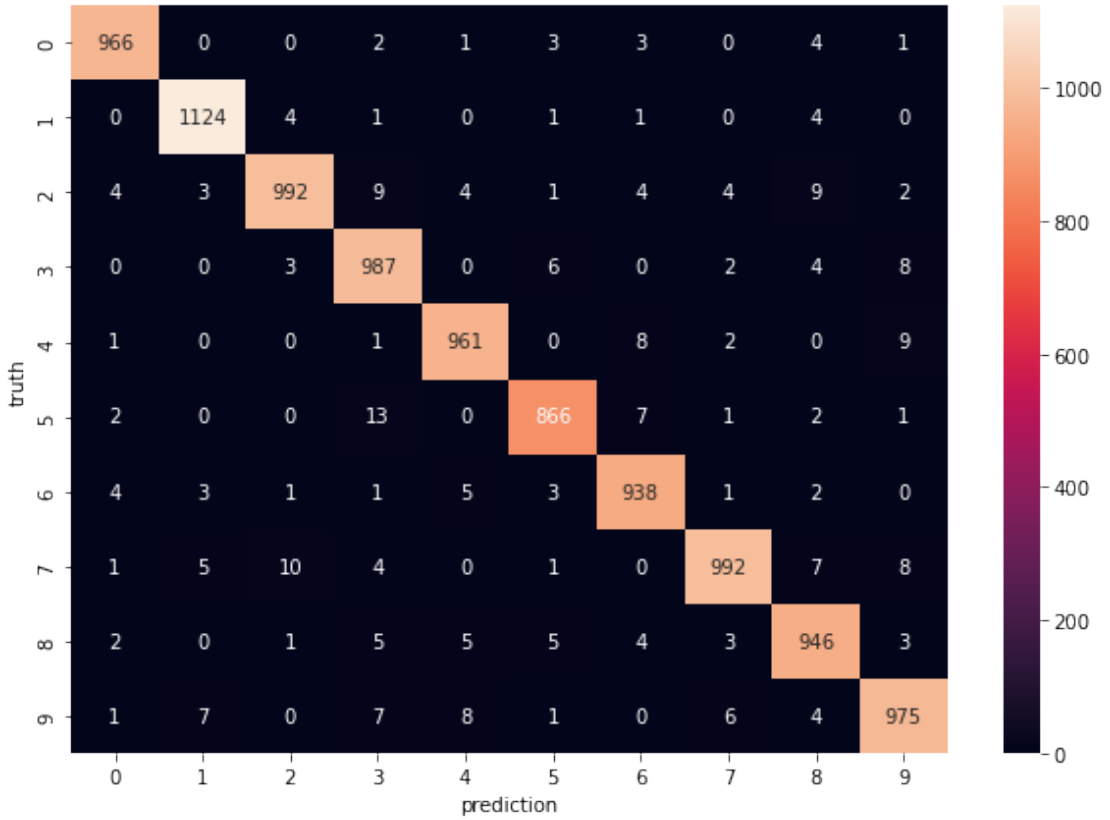


PRODUCING HIDDEN LAYERS

```
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_flat,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 3s 2ms/step - loss:
0.2781 - accuracy: 0.9202
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.1267 - accuracy: 0.9629
Epoch 3/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.0879 - accuracy: 0.9741
Epoch 4/5
1875/1875 [==============================] - 2s 1ms/step - loss:
```

```
0.0685 - accuracy: 0.9791
Epoch 5/5
1875/1875 [==============================] - 2s 1ms/step - loss:
0.0524 - accuracy: 0.9837

<keras.callbacks.History at 0x22c58559820>

model.evaluate(x_test_flat,y_test)

313/313 [==============================] - 1s 1ms/step - loss: 0.0786
- accuracy: 0.9747

[0.07858286052942276, 0.9746999740600586]

y_pred = model.predict(x_test_flat)

313/313 [==============================] - 0s 1ms/step

y_pred_labels = []
for i in range(len(y_pred)):
    y_pred_labels.append(np.argmax(y_pred[i]))
y_pred_labels[0]
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred_labels)

import seaborn as sn
plt.figure(figsize= (10,7))
sn.heatmap(cm,annot=True,fmt='d')
plt.xlabel('prediction')
plt.ylabel('truth')

Text(69.0, 0.5, 'truth')
```



CODE WITHOUT PRODUCING A FLATTEN ARRAY

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
    keras.layers.Dense(100,activation='relu'),
    keras.layers.Dense(10,activation='sigmoid')
])
model.compile(
    optimizer='RMSprop',
    loss = 'SparseCategoricalCrossentropy',
    metrics=['Hinge']
    )

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

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss:
0.2727 - hinge: 0.5250
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss:
0.1320 - hinge: 0.7874
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss:
0.0982 - hinge: 0.8886
Epoch 4/5
```

```
1875/1875 [==============================] - 3s 1ms/step - loss:
0.0813 - hinge: 0.9276
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss:
0.0701 - hinge: 0.9580

<keras.callbacks.History at 0x22c51d301f0>
```