In [ ]:

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

In [ ]:

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

In [ ]:

```
len(x_train)
```

Out[ ]:

```
60000
```

In [ ]:

```
len(x_test)
```

Out[ ]:

```
10000
```

In [ ]:

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

Out[ ]:

```
(28, 28)
```

In [ ]:

```
x_train[0]
```

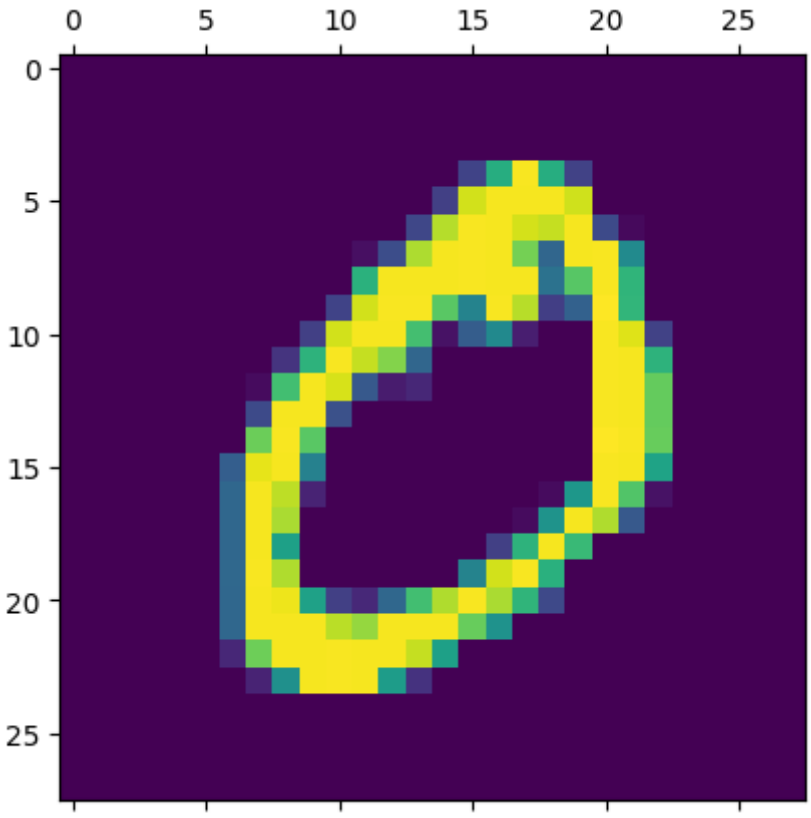
Out[ ]:

```
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,  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,
  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,  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,  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, 249, 253, 249,  64,   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,  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,  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,  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,  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,  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, 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]], dtype=uint8)
```

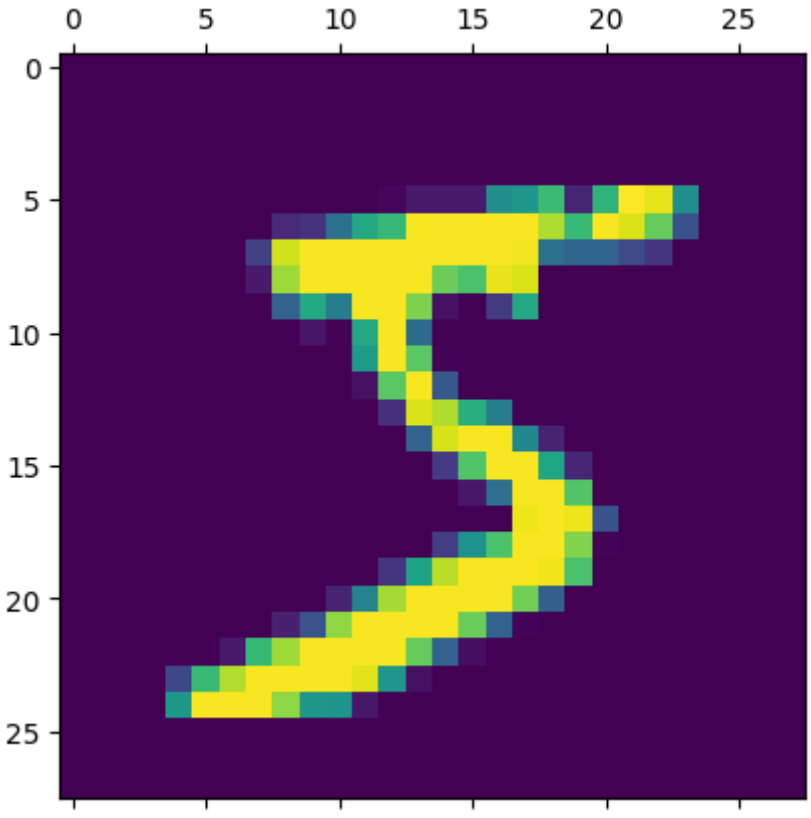
In [ ]:
```
plt.matshow(x_train[1])
```

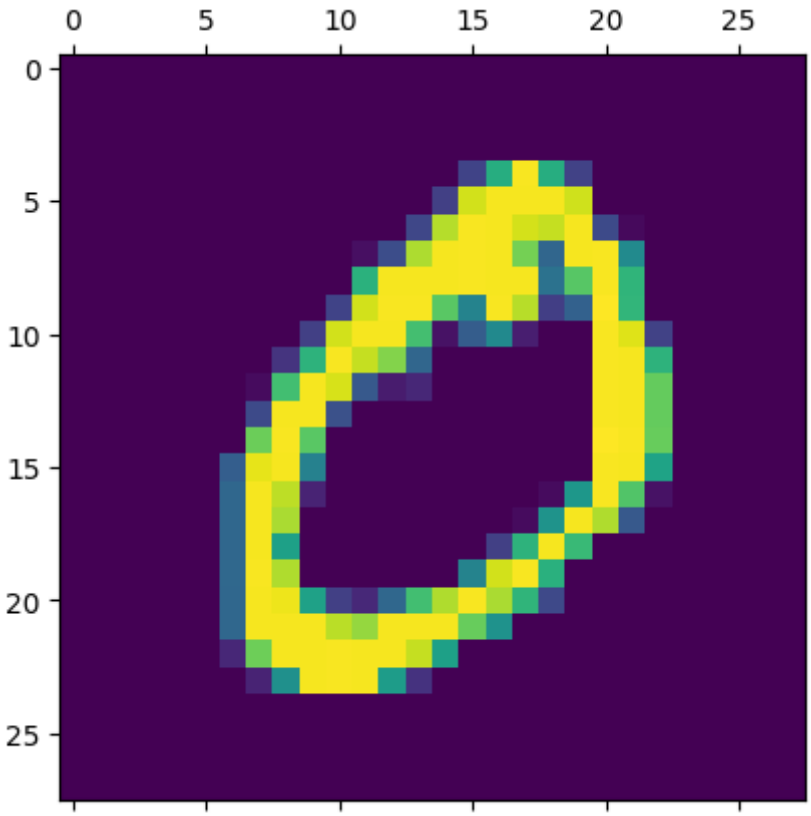
Out[ ]:
```
<matplotlib.image.AxesImage at 0x1cfa4ec6310>
```

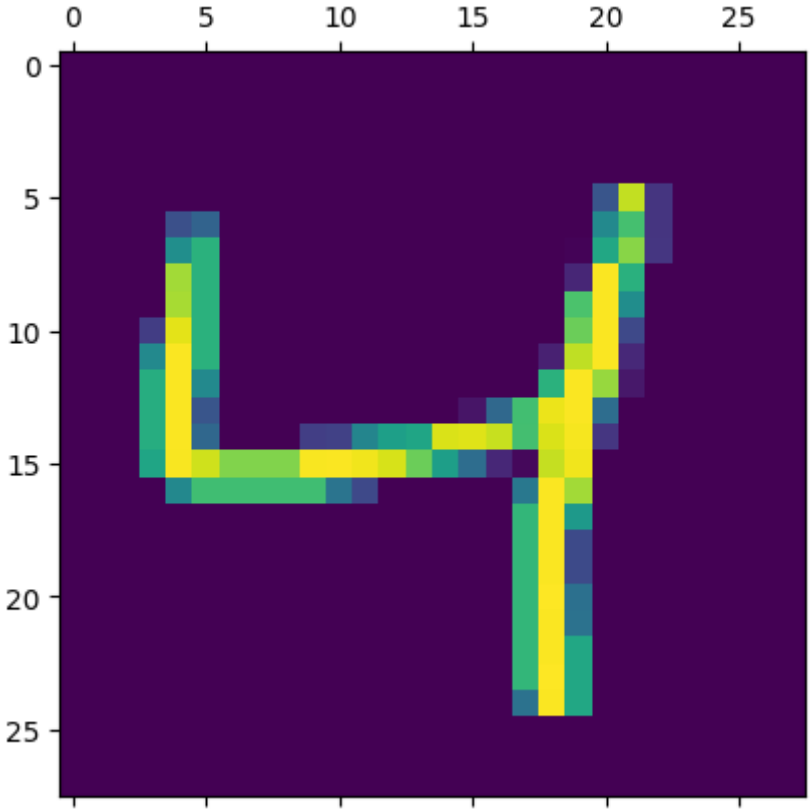


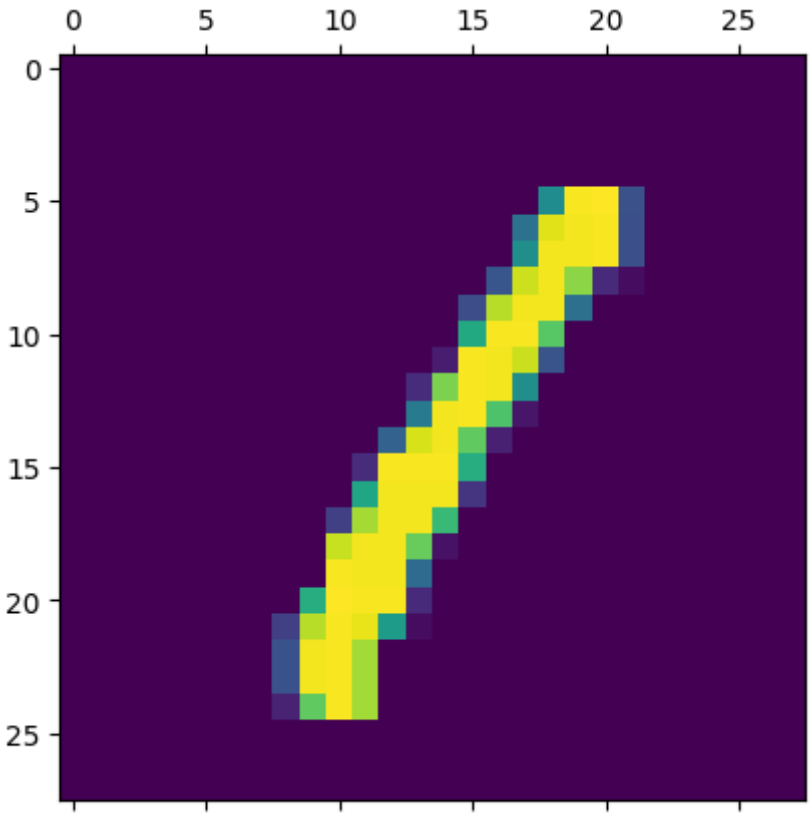
In [ ]:

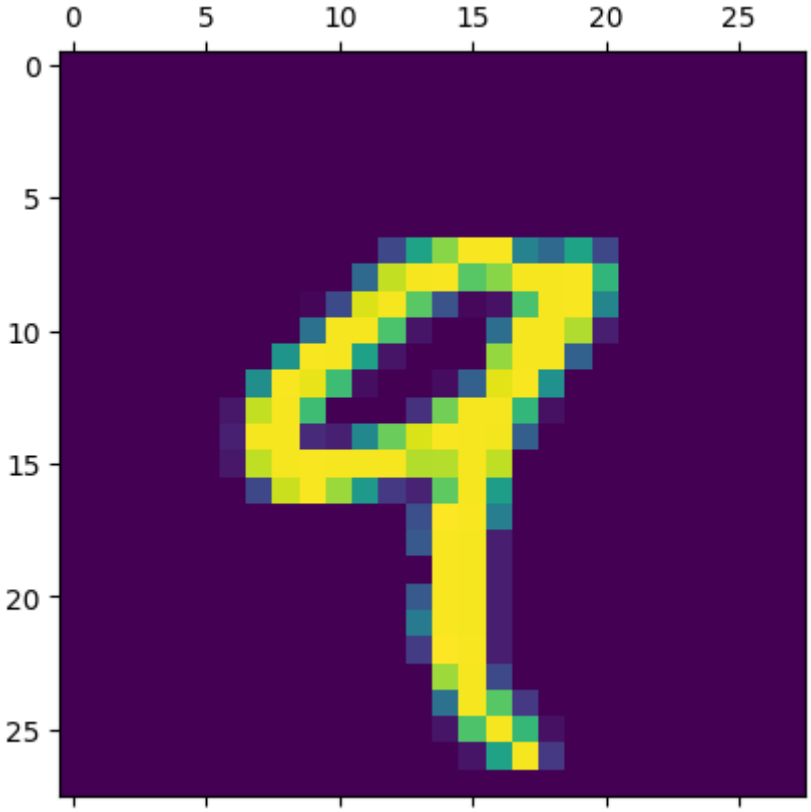
```
for i in range(10):
    plt.matshow(x_train[i])
```

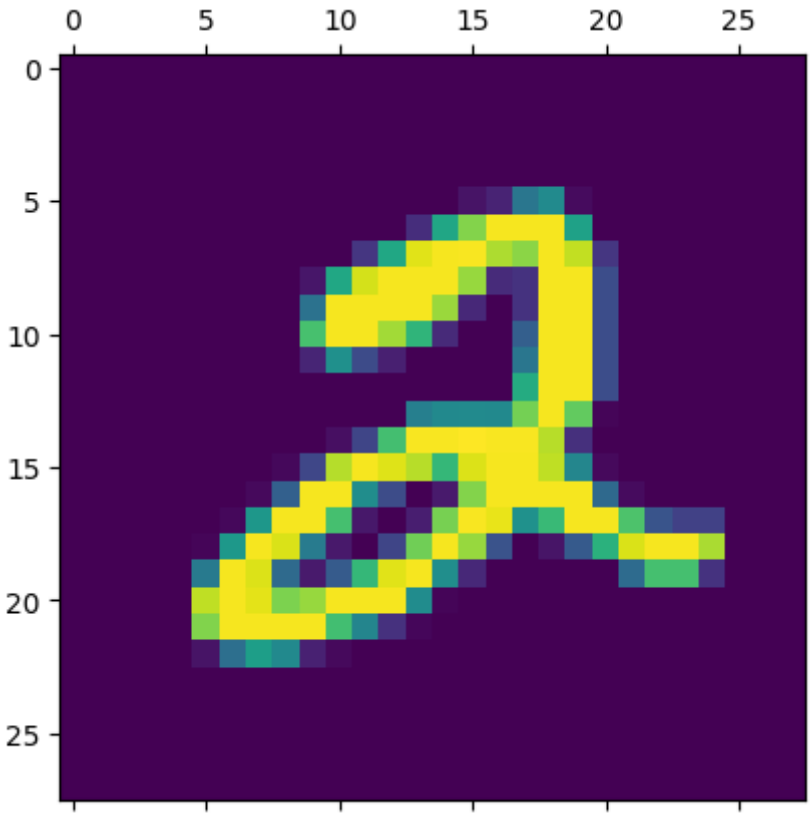


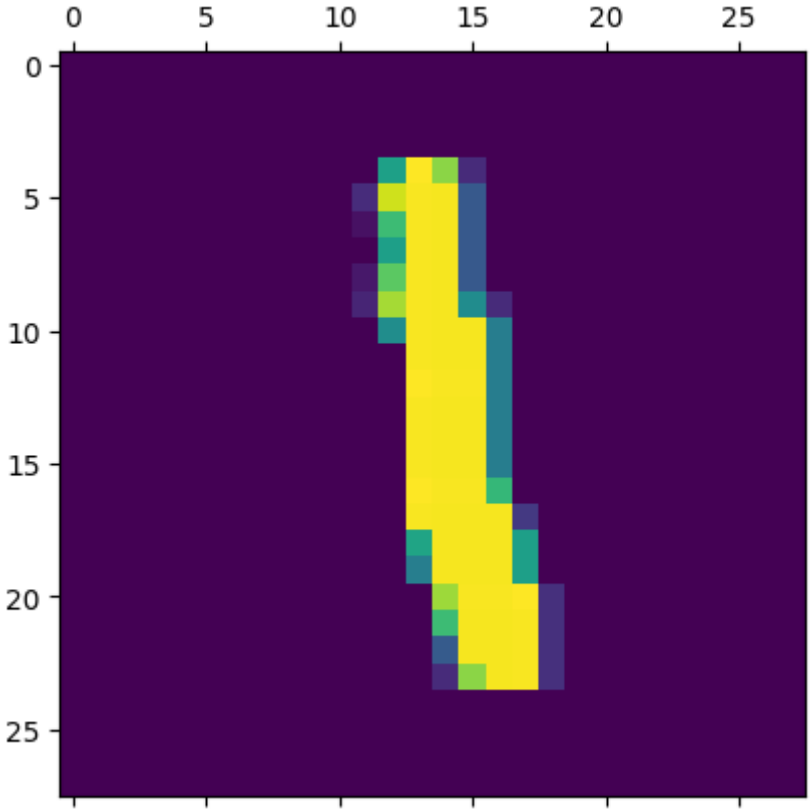

0
5
10
15
20
25
0
5
10
15
20
25

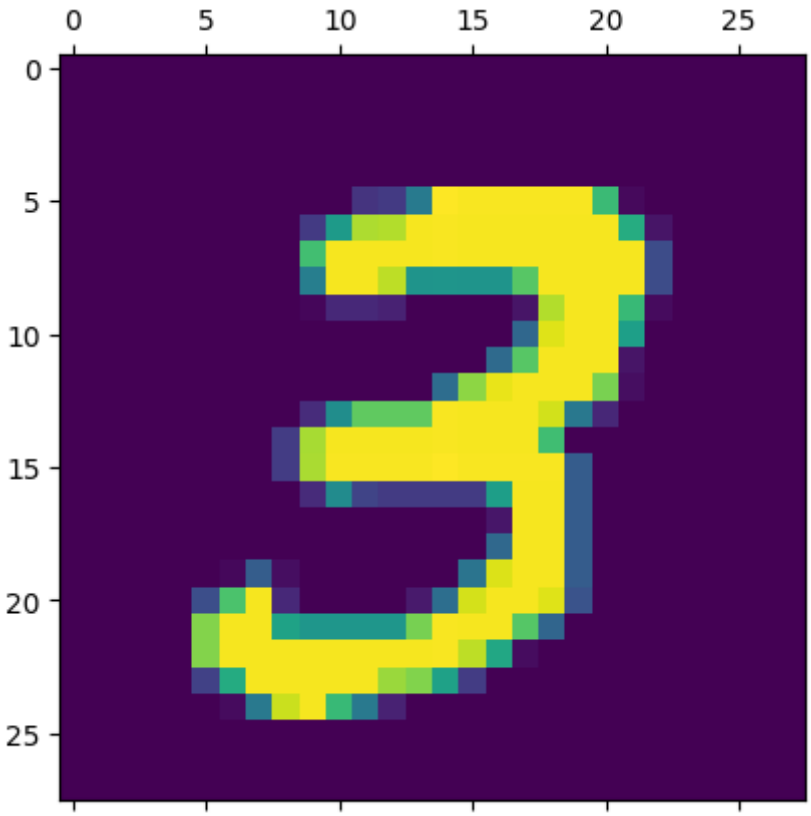

0
5
10
15
20
25
0
5
10
15
20
25

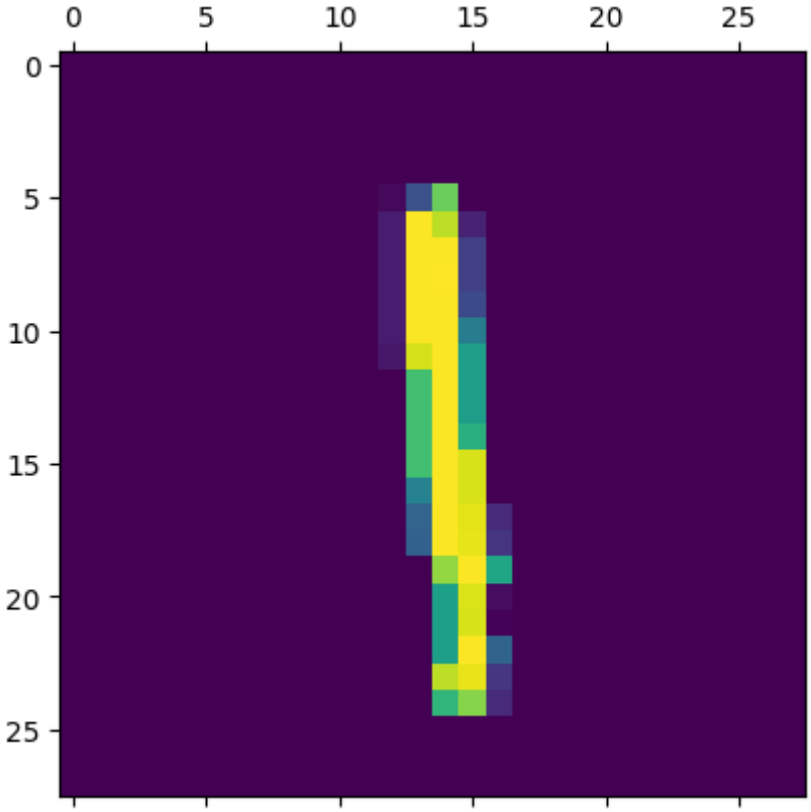

0
5
10
15
20
25
0
5
10
15
20
25

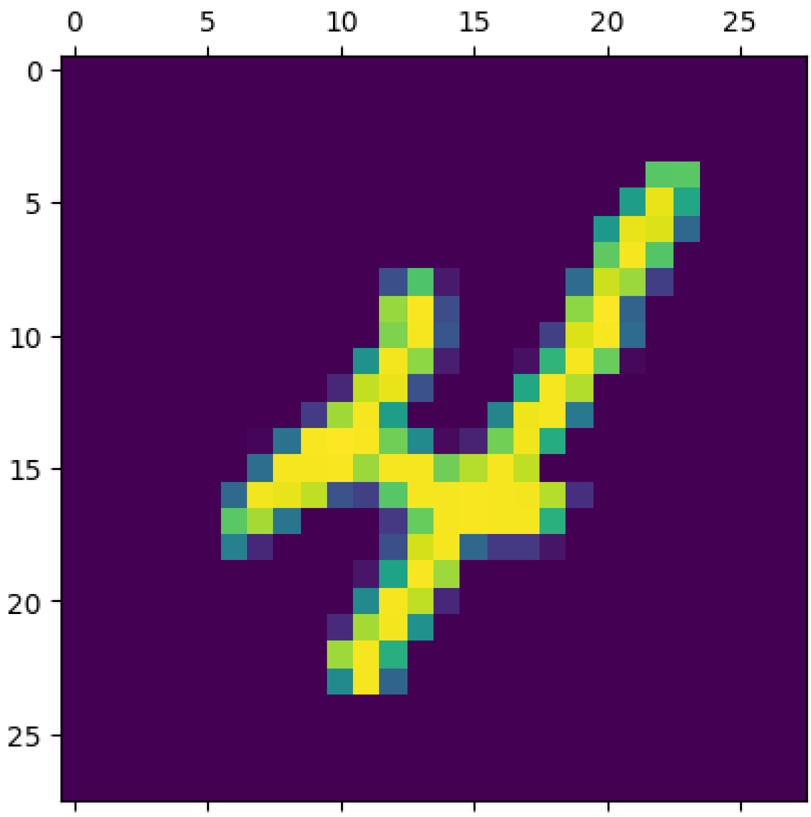

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25



In [ ]:

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

In [ ]:

```
x_train[0]
```

Out[ ]:

```
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.        , 0.01176471, 0.07058824, 0.07058824,
 0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
 0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.11764706, 0.14117647,
 0.36862745, 0.60392157, 0.66666667, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.88235294, 0.6745098 ,
 0.99215686, 0.94901961, 0.76470588, 0.25098039, 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.19215686, 0.93333333, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.98431373, 0.36470588, 0.32156863,
 0.32156863, 0.21960784, 0.15294118, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.07058824, 0.85882353, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.77647059,
 0.71372549, 0.96862745, 0.94509804, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.31372549, 0.61176471,
 0.41960784, 0.99215686, 0.99215686, 0.80392157, 0.04313725,
 0.        , 0.16862745, 0.60392157, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.05490196,
 0.00392157, 0.60392157, 0.99215686, 0.35294118, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.54509804, 0.99215686, 0.74509804, 0.00784314,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.04313725, 0.74509804, 0.99215686, 0.2745098 ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.1372549 , 0.94509804, 0.88235294,
 0.62745098, 0.42352941, 0.00392157, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.31764706, 0.94117647,
 0.99215686, 0.99215686, 0.46666667, 0.09803922, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.17647059,
 0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.0627451 , 0.36470588, 0.98823529, 0.99215686, 0.73333333,
 0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.97647059, 0.99215686, 0.97647059,
         0.25098039, 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.18039216,
         0.50980392, 0.71764706, 0.99215686, 0.99215686, 0.81176471,
         0.00784314, 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.15294118, 0.58039216, 0.89803922,
         0.99215686, 0.99215686, 0.99215686, 0.98039216, 0.71372549,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
         0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.09019608, 0.25882353,
         0.83529412, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
         0.77647059, 0.31764706, 0.00784314, 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.07058824, 0.67058824, 0.85882353, 0.99215686,
         0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,
         0.03529412, 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.21568627,
         0.6745098 , 0.88627451, 0.99215686, 0.99215686, 0.99215686,
         0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.53333333,
         0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,
         0.51764706, 0.0627451 , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ]])
```

In [ ]:

```
x_train.shape
```

Out[ ]:
```
(60000, 28, 28)
```

Modeling, Compiling & Fitting the Model

Activation Function = Sigmoid loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4679 - accuracy:
0.8783
Epoch 2/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3041 - accuracy:
0.9155
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2840 - accuracy:
0.9201
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2736 - accuracy:
0.9232
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2665 - accuracy:
0.9260
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cfb553fe50>
```

activation Function = Sigmoid optimizer = Adadelta loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='Adadelta',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 6s 3ms/step - loss: 2.3146 - accuracy:
0.1239
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 2.2282 - accuracy:
0.1761
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 2.1443 - accuracy:
0.2398
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 2.0642 - accuracy:
0.3089
Epoch 5/5
```

```
1875/1875 [==============================] - 4s 2ms/step - loss: 1.9884 - accuracy:
0.3801
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cf90b8f590>
```

activation Function = Sigmoid optimizer = Adagrad loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    a = keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='Adagrad',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 1.5136 - accuracy:
0.6484
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 1.0354 - accuracy:
0.7933
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.8766 - accuracy:
0.8184
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7887 - accuracy:
0.8314
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.7308 - accuracy:
0.8399
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cf9dd5e550>
```

activation Function = Sigmoid optimizer = Adamax loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='Adamax',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.6406 - accuracy:
0.8443
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3747 - accuracy:
0.8982
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3378 - accuracy:
0.9061
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3202 - accuracy:
0.9111
Epoch 5/5
```

```
1875/1875 [==============================] - 5s 2ms/step - loss: 0.3088 - accuracy:
0.9140
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cf9f4e1390>
```

activation Function = Sigmoid optimizer = Ftrl loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='Ftrl',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 1.4858 - accuracy:
0.7742
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 1.0181 - accuracy:
0.8249
Epoch 3/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.8619 - accuracy:
0.8381
Epoch 4/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.7757 - accuracy:
0.8464
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.7190 - accuracy:
0.8521
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cf9f7a3390>
```

activation Function = Sigmoid optimizer = Nadam loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='Nadam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 6s 2ms/step - loss: 0.4754 - accuracy:
0.8769
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3043 - accuracy:
0.9152
Epoch 3/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2839 - accuracy:
0.9203
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2732 - accuracy:
0.9239
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2668 - accuracy:
0.9254
```

Out[ ]: <keras.src.callbacks.History at 0x1cf9fdeb390>

activation Function = Sigmoid optimizer = RMSprop loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='RMSprop',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.4356 - accuracy:
0.8848
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3041 - accuracy:
0.9157
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2904 - accuracy:
0.9201
Epoch 4/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2842 - accuracy:
0.9226
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2801 - accuracy:
0.9247
```

Out[ ]: <keras.src.callbacks.History at 0x1cfa0e9b390>

activation Function = Sigmoid optimizer = SGD loss Function = sparse_categorical_crossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='SGD',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7716 - accuracy:
0.8190
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4566 - accuracy:
0.8800
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4038 - accuracy:
0.8913
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3773 - accuracy:
0.8963
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3604 - accuracy:
0.9003
```

Out[ ]: <keras.src.callbacks.History at 0x1cfa0f3b390>

The Highest Accurancy From the Optimizer : Adam : 92.60% Followed by Nadam, RMSprop

Different Loss Functions other than SparseCategoricalCrossentropy

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='adam',
              loss='MeanSquaredError',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 20.3317 - accuracy:
0.1260
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 20.2786 - accuracy:
0.1396
Epoch 3/5
1875/1875 [==============================] - 5s 3ms/step - loss: 20.2777 - accuracy:
0.1526
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 20.2775 - accuracy:
0.1587
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 20.2774 - accuracy:
0.1599
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cfb3183390>
```

In [ ]:
```
Model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(10, activation = 'sigmoid')
    ])
Model.compile(optimizer='adam',
              loss='MeanAbsoluteError',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 3.6585 - accuracy:
0.0837
Epoch 2/5
1875/1875 [==============================] - 5s 3ms/step - loss: 3.6515 - accuracy:
0.0975
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 3.6514 - accuracy:
0.1080
Epoch 4/5
1875/1875 [==============================] - 5s 3ms/step - loss: 3.6514 - accuracy:
0.1134
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 3.6514 - accuracy:
0.1141
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cfb30ab390>
```

Plotting Heatmap Based on Single Layer Neural Network

In [ ]:
```
x_train_flattened = x_train.reshape(len(x_train),28*28)
x_test_flattened = x_test.reshape(len(x_test),28*28)
```

In [ ]:
```
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)
```

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.4735 - accuracy:
0.8760
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3042 - accuracy:
0.9146
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2838 - accuracy:
0.9201
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2736 - accuracy:
0.9243
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2671 - accuracy:
0.9257
```

Out[ ]:
```
<keras.src.callbacks.History at 0x1cf00133390>
```

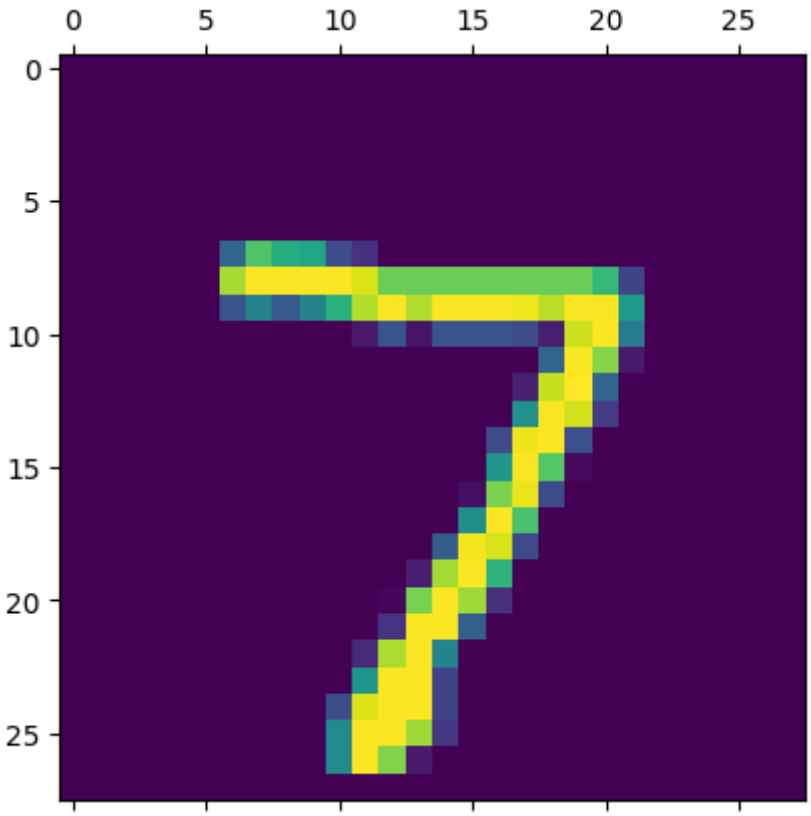
In [ ]:
```
y_predicted = Model.predict(x_test_flattened)
y_predicted[0]
```

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

Out[ ]:
```
array([2.62608863e-02, 4.75383700e-07, 6.61899745e-02, 9.59083021e-01,
       2.81140697e-03, 1.43698499e-01, 3.64029097e-06, 9.99808550e-01,
       1.17423005e-01, 7.49801993e-01], dtype=float32)
```

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

Out[ ]:
```
<matplotlib.image.AxesImage at 0x1cf00099cd0>
```



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

Out[ ]:
```
7
```

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

In [ ]:
```
y_predicted_labels[0:5]
```

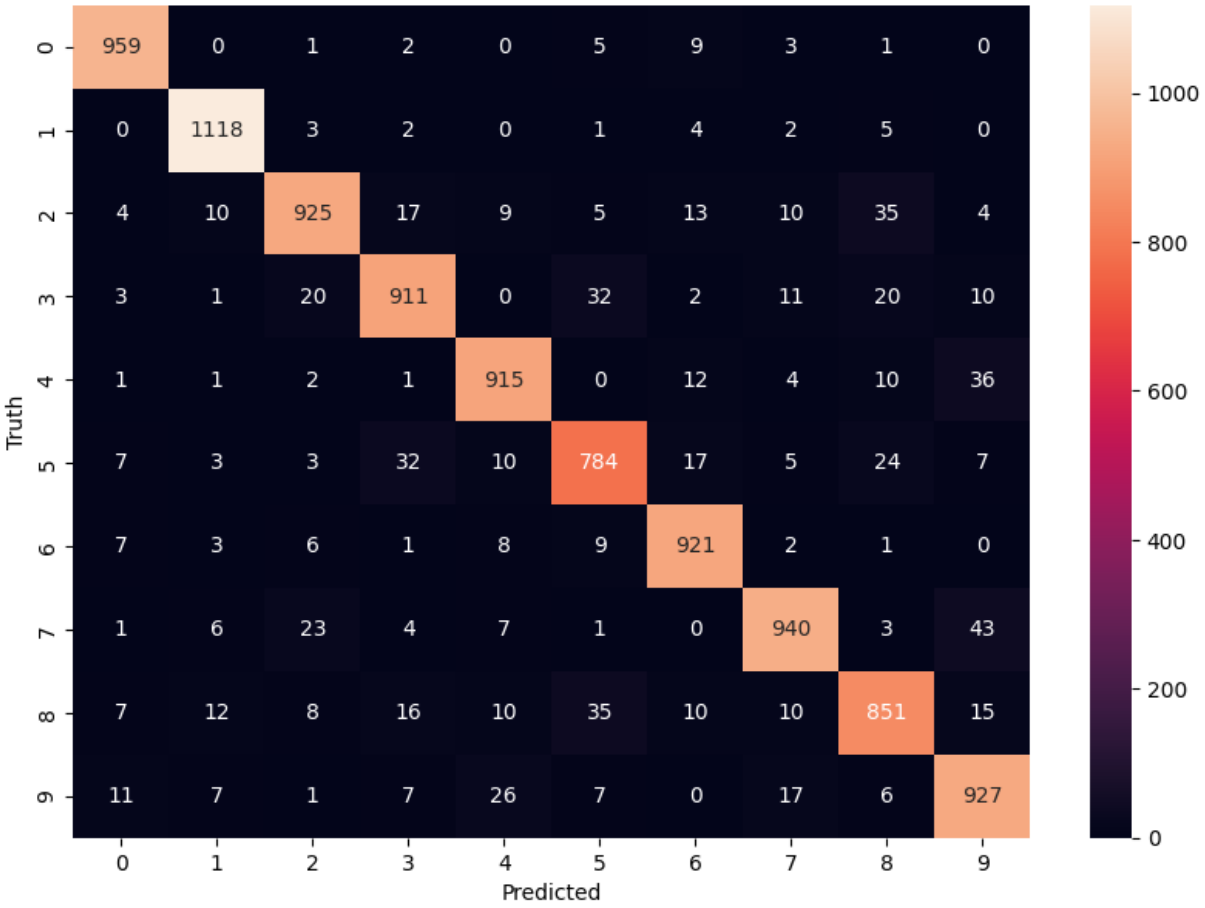
Out[ ]:
```
[7, 2, 1, 0, 4]
```

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

Out[ ]:
```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 959,    0,    1,    2,    0,    5,    9,    3,    1,    0],
       [   0, 1118,    3,    2,    0,    1,    4,    2,    5,    0],
       [   4,   10,  925,   17,    9,    5,   13,   10,   35,    4],
       [   3,    1,   20,  911,    0,   32,    2,   11,   20,   10],
       [   1,    1,    2,    1,  915,    0,   12,    4,   10,   36],
       [   7,    3,    3,   32,   10,  784,   17,    5,   24,    7],
       [   7,    3,    6,    1,    8,    9,  921,    2,    1,    0],
       [   1,    6,   23,    4,    7,    1,    0,  940,    3,   43],
       [   7,   12,    8,   16,   10,   35,   10,   10,  851,   15],
       [  11,    7,    1,    7,   26,    7,    0,   17,    6,  927]])>
```

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

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



Adding Hidden Layer in Neural Network

In [ ]:
```
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='Adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
              )

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

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2715 - accuracy:
0.9225
Epoch 2/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.1241 - accuracy:
0.9629
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0874 - accuracy:
0.9738
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0656 - accuracy:
0.9801
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0522 - accuracy:
0.9841
```

Out[ ]: <keras.src.callbacks.History at 0x1cf01dc4790>

The Accuracy in Single Neural Network was 92.60% The Accuracy in Neural Network containing Hidden Layers is 98.41%

In [ ]:
```
Model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0880 - accuracy: 0.
9736
```

Out[ ]:
```
[0.08802524209022522, 0.9735999703407288]
```