

```
import tensorflow as tf
from tensorflow import keras
```

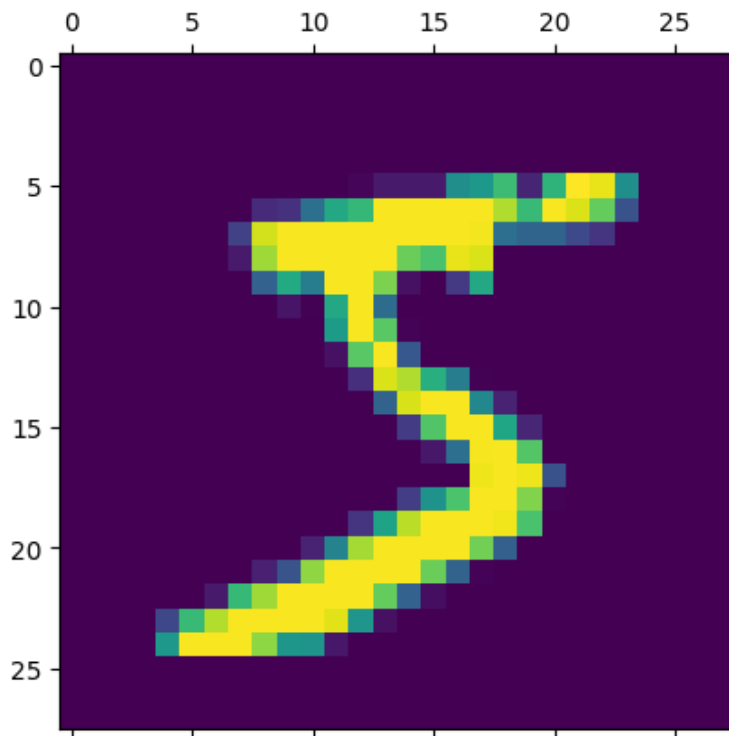
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

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

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>  
11490434/11490434 ————— 1s 0us/step

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

<matplotlib.image.AxesImage at 0x7fd13db12120>



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

```
x_train[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.      ]]

```

```

model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(128,activation='relu'),
    keras.layers.Dense(10,activation='softmax')
])

```

```

/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not
super().__init__(**kwargs)

```

```

model.summary()

```

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100,480
dense_1 (Dense)	(None, 10)	1,290

Total params: 101,770 (397.54 KB)

Trainable params: 101,770 (397.54 KB)

Non-trainable params: 0 (0.00 B)

```
model.compile(optimizer='sgd',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

```
history = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs = 10)
```

```
Epoch 1/10
1875/1875 ————— 7s 3ms/step - accuracy: 0.7370 - loss: 1.0084 - val_accuracy: 0.9044
Epoch 2/10
1875/1875 ————— 5s 3ms/step - accuracy: 0.9027 - loss: 0.3548 - val_accuracy: 0.9179
Epoch 3/10
1875/1875 ————— 6s 3ms/step - accuracy: 0.9156 - loss: 0.3004 - val_accuracy: 0.9272
Epoch 4/10
1875/1875 ————— 6s 3ms/step - accuracy: 0.9250 - loss: 0.2649 - val_accuracy: 0.9310
Epoch 5/10
1875/1875 ————— 6s 3ms/step - accuracy: 0.9291 - loss: 0.2516 - val_accuracy: 0.9347
Epoch 6/10
1875/1875 ————— 10s 3ms/step - accuracy: 0.9381 - loss: 0.2244 - val_accuracy: 0.9405
Epoch 7/10
1875/1875 ————— 9s 3ms/step - accuracy: 0.9411 - loss: 0.2096 - val_accuracy: 0.9428
Epoch 8/10
1875/1875 ————— 6s 3ms/step - accuracy: 0.9445 - loss: 0.1977 - val_accuracy: 0.9477
Epoch 9/10
1875/1875 ————— 5s 3ms/step - accuracy: 0.9485 - loss: 0.1829 - val_accuracy: 0.9500
Epoch 10/10
1875/1875 ————— 6s 3ms/step - accuracy: 0.9525 - loss: 0.1692 - val_accuracy: 0.9502
```

```
test_loss,test_acc = model.evaluate(x_test,y_test)
print("Loss = %.3f" %test_loss)
print("Accuracy = %.3f" %test_acc)
```

```
313/313 ————— 1s 2ms/step - accuracy: 0.9418 - loss: 0.1915
Loss = 0.166
Accuracy = 0.950
```

```
n = random.randint(0,9999)
plt.imshow(x_test[n])
plt.show
```

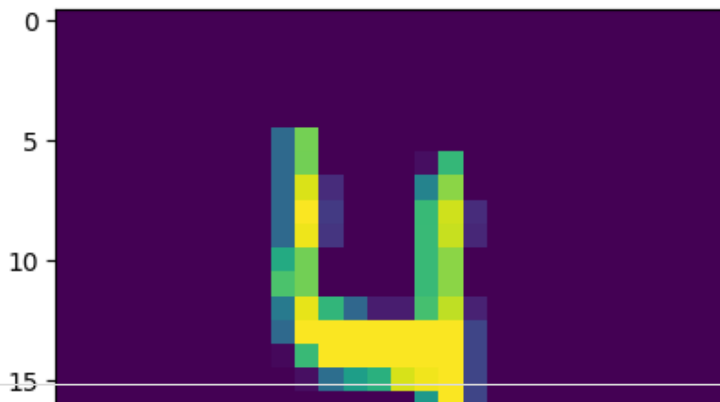
```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None
```

</usr/local/lib/python3.12/dist-packages/matplotlib/pyplot.py>  
Display all open figures.

Parameters

-----

block : bool, optional



```
test_predict = model.predict(x_test)
test_predict_labels = np.argmax(test_predict,axis = 1)
confusion_matrix = tf.math.confusion_matrix(labels=y_test,predictions=test_predict_labels)
print('Confusion Matrix of the test set :\n', confusion_matrix)
```

313/313 ----- 1s 2ms/step

Confusion Matrix of the test set :

tf.Tensor(

```
[[ 963  0  51  1 10  0  45  7 20  2 251]
 [  0 1116  2  2  0  1  4  2  8  0]
 [  7  1 976  7  7  3  9 10 11  1]
 [  0  1 13 949  0 21  1 12 10  3]
 [  1  1  4  0 935  0 12  3  3 23]
 [  9  1  0 13  4 837 13  2  7  6]
 [  8  3  5  0 10 12 918  1  1  0]
 [  1  8 22  4  6  2  0 965  2 18]
 [  4  2  4 14  8 11 12  8 905  6]
 [  9  6  1  7 25  5  2 11  5 938]], shape=(10, 10), dtype=int32)
```

Start coding or [generate](#) with AI.