import tensorflow as tf from tensorflow import keras

**import** tensorflow **as** tf

**from** tensorflow **import** keras

C:\Users\LENOVO\AppData\Roaming\Python\Python39\site-packages\scipy\\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.17.3 and <1.25.0 is required for this version of SciPy (detected version 1.26.1

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

In [2]:

mnist\_dataset **=** tf.keras.datasets.mnist

In [3]:

(x\_train, y\_train), (x\_test, y\_test) **=** mnist\_dataset.load\_data()

In [4]:

len(x\_train)

Out[4]:

60000

In [5]:

x\_train.shape

Out[5]:

(60000, 28, 28)

In [6]:

**import** matplotlib.pyplot **as** plt

In [7]:

plt.matshow(x\_train[2])

Out[7]:

<matplotlib.image.AxesImage at 0x26092265af0>

A pixelated image of a number

Description automatically generated

In [8]:

x\_train **=** x\_train**/**255

x\_test **=** x\_test**/**255

In [9]:

x\_train[0]

Out[9]:

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 [10]:

model **=** keras.Sequential([ *# he model starts with a Flatten layer to prepare the input.*

keras.layers.Flatten(input\_shape**=**(28, 28)),

keras.layers.Dense(128, activation**=**'relu'),

keras.layers.Dense(10, activation**=**'softmax')

])

In [11]:

model.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

flatten (Flatten) (None, 784) 0

dense (Dense) (None, 128) 100480

dense\_1 (Dense) (None, 10) 1290

=================================================================

Total params: 101770 (397.54 KB)

Trainable params: 101770 (397.54 KB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [12]:

model.compile(optimizer**=**'sgd', *# stochastic gradient descent*

loss**=**'sparse\_categorical\_crossentropy', *# Use this crossentropy loss function when there are*

*#two or more label classes. We expect labels to be provided as integers, mutually exclusive events*

metrics**=**['accuracy']) *# parameter is accuracy*

In [13]:

history **=** model.fit(x\_train, y\_train, validation\_data**=**(x\_test, y\_test), epochs**=**10)

Epoch 1/10

1875/1875 [==============================] - 17s 7ms/step - loss: 0.6548 - accuracy: 0.8356 - val\_loss: 0.3617 - val\_accuracy: 0.9013

Epoch 2/10

1875/1875 [==============================] - 11s 6ms/step - loss: 0.3392 - accuracy: 0.9054 - val\_loss: 0.2968 - val\_accuracy: 0.9162

Epoch 3/10

1875/1875 [==============================] - 12s 6ms/step - loss: 0.2885 - accuracy: 0.9191 - val\_loss: 0.2624 - val\_accuracy: 0.9270

Epoch 4/10

1875/1875 [==============================] - 12s 6ms/step - loss: 0.2573 - accuracy: 0.9283 - val\_loss: 0.2386 - val\_accuracy: 0.9325

Epoch 5/10

1875/1875 [==============================] - 12s 6ms/step - loss: 0.2338 - accuracy: 0.9340 - val\_loss: 0.2223 - val\_accuracy: 0.9351

Epoch 6/10

1875/1875 [==============================] - 12s 6ms/step - loss: 0.2149 - accuracy: 0.9401 - val\_loss: 0.2043 - val\_accuracy: 0.9405

Epoch 7/10

1875/1875 [==============================] - 13s 7ms/step - loss: 0.1992 - accuracy: 0.9441 - val\_loss: 0.1905 - val\_accuracy: 0.9443

Epoch 8/10

1875/1875 [==============================] - 13s 7ms/step - loss: 0.1858 - accuracy: 0.9473 - val\_loss: 0.1805 - val\_accuracy: 0.9471

Epoch 9/10

1875/1875 [==============================] - 12s 6ms/step - loss: 0.1746 - accuracy: 0.9507 - val\_loss: 0.1706 - val\_accuracy: 0.9511

Epoch 10/10

1875/1875 [==============================] - 11s 6ms/step - loss: 0.1644 - accuracy: 0.9534 - val\_loss: 0.1648 - val\_accuracy: 0.9527

In [14]:

test\_loss, test\_acc **=** model.evaluate(x\_test, y\_test)

print("Loss=%.3f"**%**test\_loss)

print("Accuracy=%.3f"**%**test\_acc)

313/313 [==============================] - 1s 4ms/step - loss: 0.1648 - accuracy: 0.9527

Loss=0.165

Accuracy=0.953

In [15]:

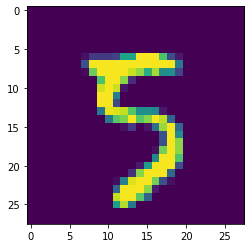
**import** random

In [16]:

n **=** random.randint(0,9999)

plt.imshow(x\_test[n])

plt.show()



In [17]:

**import** numpy **as** np

predicted\_value**=**model.predict(x\_test)

print("Handwritten number in the image is= %d" **%**np.argmax(predicted\_value[n]))

313/313 [==============================] - 1s 3ms/step

Handwritten number in the image is= 5

In [18]:

history.history.keys()

Out[18]:

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

**Training Accuracy**

In [19]:

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['Train', 'Validation'], loc**=**'upper left')

plt.show()

A graph of a line

Description automatically generated with medium confidence

**Training Loss**

In [20]:

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['Train', 'Validation'], loc**=**'upper left')

plt.show()

