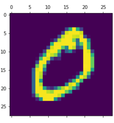
In [ ]:
```
##Name:AnmolDhar
##Roll no:36113
##Subject:LP-IV(DL)
```

In [ ]:
```
!pip install tensorflow
```

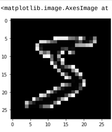
In [1]:
```
#importing necessary libraries
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

In [2]:
```
#import dataset and split into train and test data
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [3]:
```
plt.matshow(x_train[1])
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x216d2a9c9d0>
```


In [4]:
```
plt.imshow(-x_train[0], cmap="gray")
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x216d321cac0>
```


In [5]:
```
x_train = x_train / 255
x_test = x_test / 255
```

In [6]:
```
model = keras.Sequential([
keras.layers.Flatten(input_shape=(28, 28)),
keras.layers.Dense(128, activation="relu"),
keras.layers.Dense(10, activation="softmax")
])

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: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

In [7]:
```
model.compile(optimizer="sgd",
loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

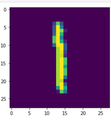
In [8]:
```
history=model.fit(x_train,
y_train,validation_data=(x_test,y_test),epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.6614 - accuracy: 0.8326 - val_loss: 0.3559 - val_accuracy: 0.9034
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3358 - accuracy: 0.9069 - val_loss: 0.2906 - val_accuracy: 0.9198
Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2860 - accuracy: 0.9197 - val_loss: 0.2582 - val_accuracy: 0.9290
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2551 - accuracy: 0.9286 - val_loss: 0.2334 - val_accuracy: 0.9348
Epoch 5/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2325 - accuracy: 0.9351 - val_loss: 0.2150 - val_accuracy: 0.9390
Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2140 - accuracy: 0.9397 - val_loss: 0.2011 - val_accuracy: 0.9437
Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1991 - accuracy: 0.9441 - val_loss: 0.1890 - val_accuracy: 0.9472
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1861 - accuracy: 0.9479 - val_loss: 0.1787 - val_accuracy: 0.9484
Epoch 9/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1747 - accuracy: 0.9509 - val_loss: 0.1690 - val_accuracy: 0.9505
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1649 - accuracy: 0.9537 - val_loss: 0.1604 - val_accuracy: 0.9537
```

In [9]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.1604 - accuracy: 0.9537
Loss=0.160
Accuracy=0.954
```

In [10]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```



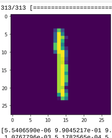
In [11]:
```
x_train
```

Out[11]:
```
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., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [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 [12]:
```
x_test
```

Out[12]:
```
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., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]])
```

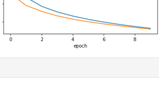
In [13]:
```
predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()

print(predicted_value[n])
```

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


```
[5.5406590e-06 9.9045217e-01 9.6445804e-04 1.1881288e-03 5.5083026e-05
 1.0767756e-03 5.1782565e-04 9.0224096e-04 4.8863539e-03 3.5130000e-04]
```

In [14]:
```
# history.history()
history.history.keys()
# dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

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()
```

In [15]:
```
# history.history()
history.history.keys()
# dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

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()
```



In [ ]: