In [2]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, Ma
import cv2
```

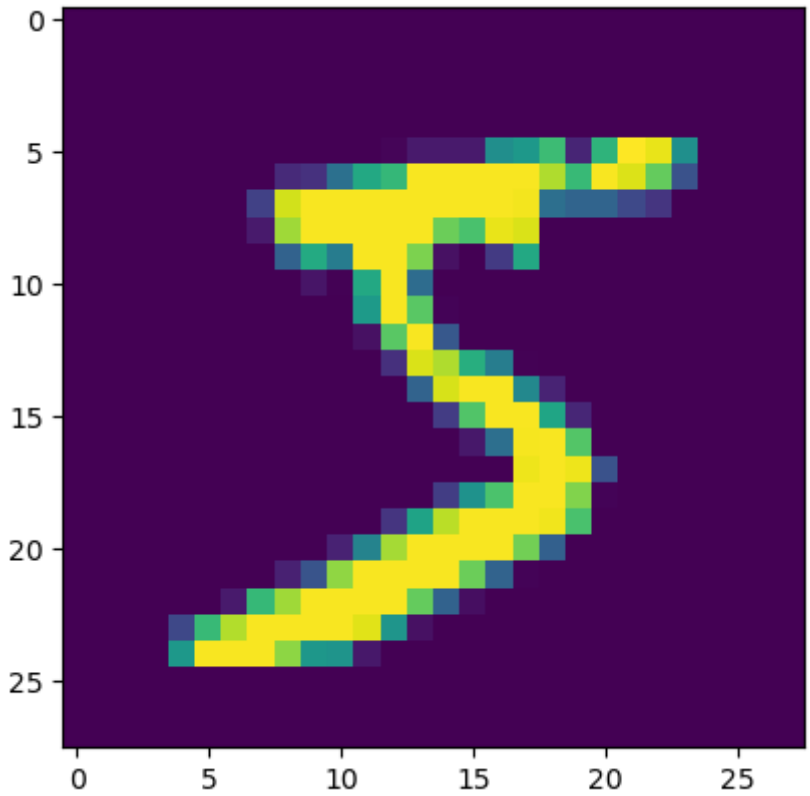
In [3]:
```
mnist = tf.keras.datasets.mnist
```

In [5]:
```
(x_train, y_train), (x_test, y_test)= mnist.load_data()
```

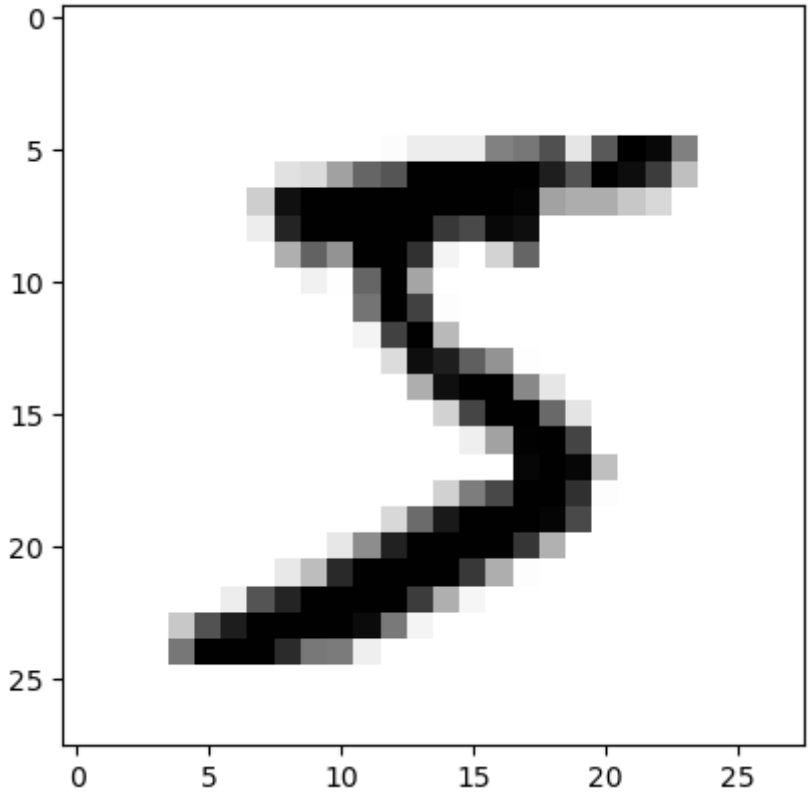
In [6]:
```
x_train.shape
```

Out[6]:
```
(60000, 28, 28)
```

In [8]:
```
plt.imshow(x_train[0])
plt.show()
plt.imshow(x_train[0], cmap=plt.cm.binary)
```



Out[8]:
```
<matplotlib.image.AxesImage at 0x214daf646d0>
```



In [9]:
```
print (x_train[0])
```

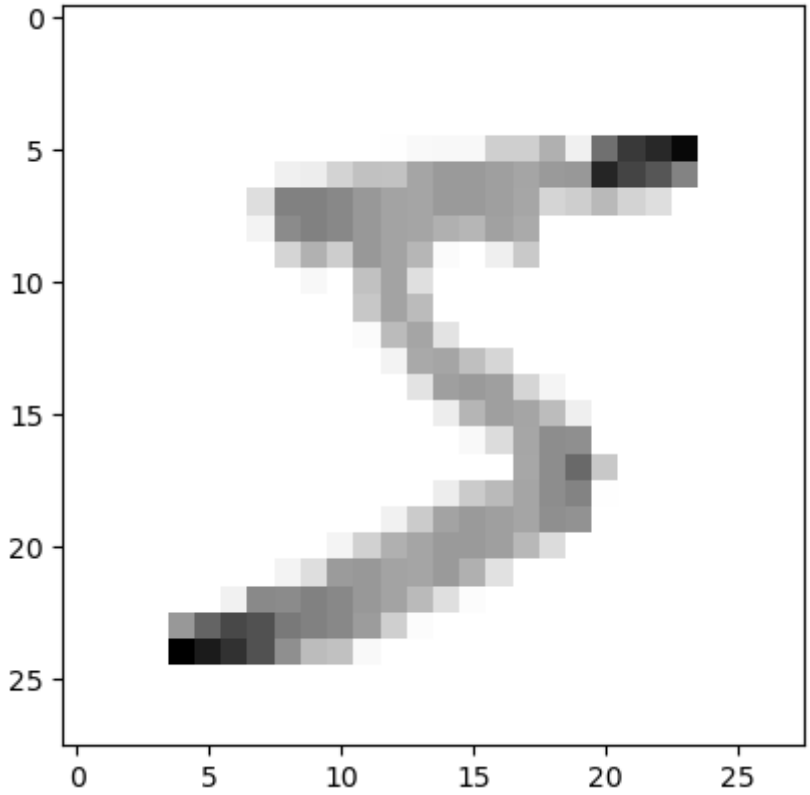
```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   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]]
```

In [10]:

```
x_train = tf.keras.utils.normalize (x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
plt.imshow(x_train[0], cmap = plt.cm.binary)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x214dafdd1e0>
```



In [11]:
```
print(x_train[0])
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.00393124 0.02332955 0.02620568 0.02625207 0.17420356 0.17566281
  0.28629534 0.05664824 0.51877786 0.71632322 0.77892406 0.89301644
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.05780486 0.06524513 0.16128198 0.22713296
  0.22277047 0.32790981 0.36833534 0.3689874  0.34978968 0.32678448
  0.368094   0.3747499  0.79066747 0.67980478 0.61494005 0.45002403
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.12250613 0.45858525 0.45852825 0.43408872 0.37314701
  0.33153488 0.32790981 0.36833534 0.3689874  0.34978968 0.32420121
  0.15214552 0.17865984 0.25626376 0.1573102  0.12298801 0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.04500225 0.4219755  0.45852825 0.43408872 0.37314701
  0.33153488 0.32790981 0.28826244 0.26543758 0.34149427 0.31128482
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.1541463  0.28272888 0.18358693 0.37314701
  0.33153488 0.26569767 0.01601458 0.         0.05945042 0.19891229
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.0253731  0.00171577 0.22713296
  0.33153488 0.11664776 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.20500962
  0.33153488 0.24625638 0.00291174 0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.01622378
  0.24897876 0.32790981 0.10191096 0.         0.         0.
  0.         0.         0.         0.         0.         0.
```

```
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.04586451 0.31235677 0.32757096 0.23335172 0.14931733 0.00129164
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.10498298 0.34940902 0.3689874  0.34978968 0.15370495
 0.04089933 0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.06551419 0.27127137 0.34978968 0.32678448
 0.245396   0.05882702 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.02333517 0.12857881 0.32549285
 0.41390126 0.40743158 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.32161793
 0.41390126 0.54251585 0.20001074 0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.06697006 0.18959827 0.25300993 0.32678448
 0.41390126 0.45100715 0.00625034 0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.05110617 0.19182076 0.33339444 0.3689874  0.34978968 0.32678448
 0.40899334 0.39653769 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.04117838 0.16813739
 0.28960162 0.32790981 0.36833534 0.3689874  0.34978968 0.25961929
 0.12760592 0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.04431706 0.11961607 0.36545809 0.37314701
 0.33153488 0.32790981 0.36833534 0.28877275 0.111988   0.00258328
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.05298497 0.42752138 0.4219755  0.45852825 0.43408872 0.37314701
 0.33153488 0.25273681 0.11646967 0.01312603 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.37491383 0.56222061
 0.66525569 0.63253163 0.48748768 0.45852825 0.43408872 0.359873
 0.17428513 0.01425695 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.92705966 0.82698729
 0.74473314 0.63253163 0.4084877  0.24466922 0.22648107 0.02359823
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```

```
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         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]:
```
print(y_train[0])
```

```
5
```

In [14]:
```
IMG_SIZE=28
x_trainr= np.array(x_train).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
x_testr=np.array(x_test).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
print("Training Samples Dimensions", x_trainr.shape)
print("Testing Samples Dimension", x_testr.shape)
```

```
Training Samples Dimensions (60000, 28, 28, 1)
Testing Samples Dimension (10000, 28, 28, 1)
```

In [18]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, Ma
```

In [22]:
```
###Creating a Neural Network
model= Sequential()

### First Convolution Layer
model.add(Conv2D(64, (3,3), input_shape = x_trainr.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

### Second Convolution Layer
model.add(Conv2D(64, (3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))


### Third Convolution Layer
model.add(Conv2D(64, (3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

##Fully Connected Layer 1
model.add (Flatten())
model.add(Dense(64))
model.add(Activation("relu"))

###Fully Connected Layer
model.add(Dense(32))
model.add(Activation("relu"))

##Last Fully connected Layer
model.add(Dense(10))
model.add(Activation('softmax'))
```

In [23]:
```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 26, 26, 64)        640

 activation_5 (Activation)   (None, 26, 26, 64)        0

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 11, 11, 64)        36928

 activation_6 (Activation)   (None, 11, 11, 64)        0

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_7 (Conv2D)           (None, 3, 3, 64)          36928

 activation_7 (Activation)   (None, 3, 3, 64)          0

 max_pooling2d_4 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 activation_8 (Activation)   (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 activation_9 (Activation)   (None, 32)                0

 dense_2 (Dense)             (None, 10)                330

 activation_10 (Activation)  (None, 10)                0

=================================================================
Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0
_________________________________________________________________
```

In [24]:
```
print ("Total Training Samples = ",len(x_trainr))
```

```
Total Training Samples =  60000
```

In [25]:
```
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=['
```

In [26]:
```
model.fit(x_trainr, y_train, epochs=5, validation_split = 0.3)
```

```
Epoch 1/5
1313/1313 [==============================] - 35s 25ms/step - loss: 0.3251 - accura
cy: 0.8971 - val_loss: 0.1317 - val_accuracy: 0.9584
Epoch 2/5
1313/1313 [==============================] - 35s 27ms/step - loss: 0.0997 - accura
cy: 0.9694 - val_loss: 0.0878 - val_accuracy: 0.9716
Epoch 3/5
1313/1313 [==============================] - 37s 28ms/step - loss: 0.0730 - accura
cy: 0.9775 - val_loss: 0.0778 - val_accuracy: 0.9768
Epoch 4/5
1313/1313 [==============================] - 38s 29ms/step - loss: 0.0575 - accura
cy: 0.9825 - val_loss: 0.0680 - val_accuracy: 0.9806
Epoch 5/5
1313/1313 [==============================] - 44s 33ms/step - loss: 0.0475 - accura
cy: 0.9849 - val_loss: 0.0627 - val_accuracy: 0.9803
```

Out[26]:
```
<keras.callbacks.History at 0x214dfd1fa60>
```

In [27]:
```
test_loss, test_acc=model.evaluate(x_testr, y_test)
print("Test loss on 10,000 test samples",test_loss)
print("Validation Accuracy on 10,000 test samples", test_acc)
```

```
313/313 [==============================] - 5s 15ms/step - loss: 0.0595 - accuracy:
0.9816
Test loss on 10,000 test samples 0.05948447808623314
Validation Accuracy on 10,000 test samples 0.9815999865531921
```

In [28]:
```
predicions = model.predict([x_testr])
```

```
313/313 [==============================] - 5s 14ms/step
```

In [29]:
```
print (predicions)
```

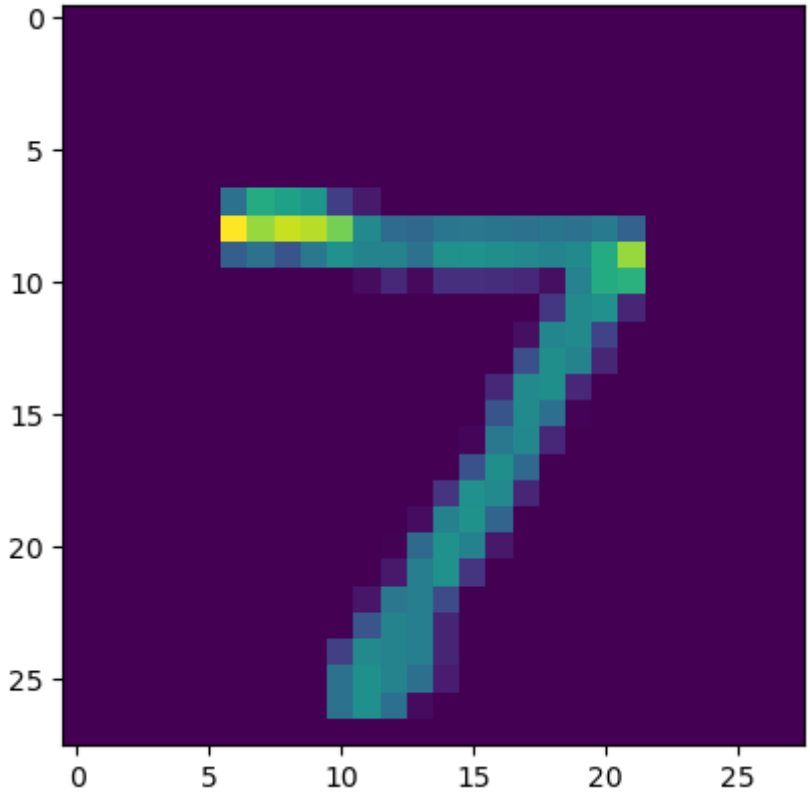
```
[[2.3403625e-07 1.9757426e-04 1.3942330e-05 ... 9.9977404e-01
  2.5197298e-07 3.6511976e-06]
 [8.3195284e-04 6.0948783e-05 9.9884486e-01 ... 2.2490613e-05
  2.8961729e-05 4.5134697e-05]
 [1.1027138e-06 9.9996841e-01 7.1710451e-06 ... 1.2037690e-05
  1.3217960e-06 5.5885989e-06]
 ...
 [1.3817364e-09 1.6167731e-07 9.0123216e-09 ... 7.6231714e-09
  8.5571191e-05 6.4584792e-06]
 [4.5747124e-06 1.7333035e-07 2.8283870e-07 ... 6.0344985e-10
  2.7017563e-04 1.4373812e-07]
 [7.1893021e-04 3.0323406e-05 1.7454075e-04 ... 9.5261427e-08
  1.9517609e-04 1.3109346e-05]]
```

In [30]:
```
print (np.argmax(predicions[0]))
```

```
7
```

In [31]:
```
plt.imshow(x_test[0])
```

Out[31]:
```
<matplotlib.image.AxesImage at 0x214e01576d0>
```

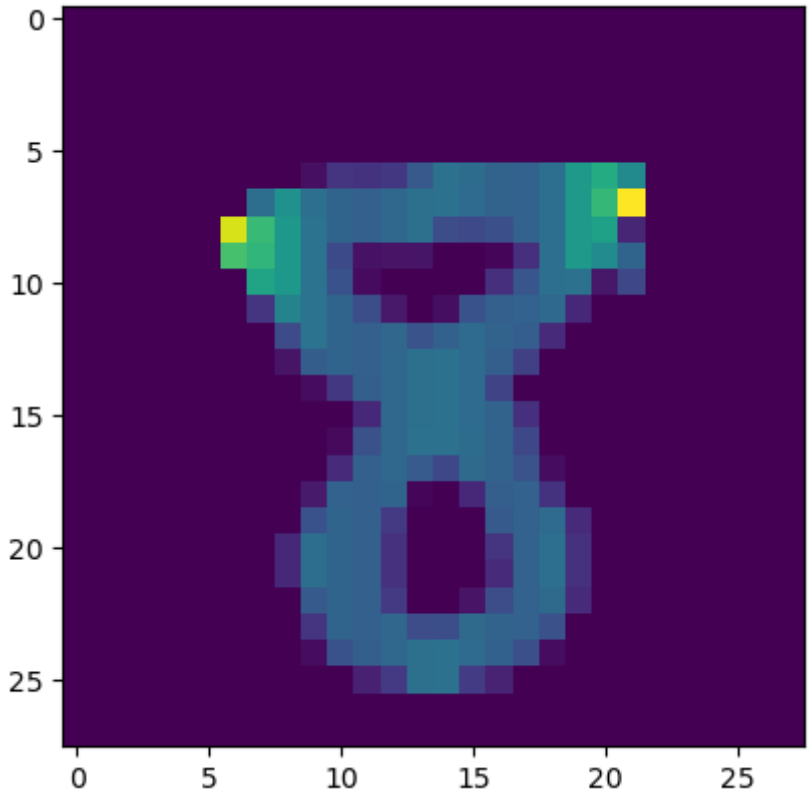


In [32]:
```
print (np.argmax(predicions[128]))
```

```
8
```

In [33]:
```
plt.imshow(x_test[128])
```

Out[33]:
```
<matplotlib.image.AxesImage at 0x214e019b160>
```



In [39]:
```
import cv2
```

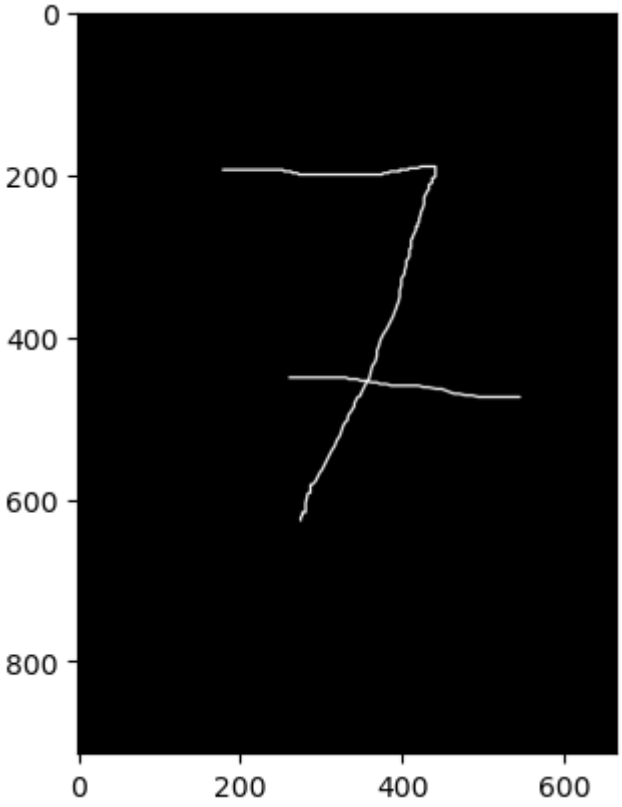
In [45]:
```
img = cv2.imread("C:/Users/DELL\OneDrive/Desktop/Handwritten Digit/digit7.png")
```

In [46]:
```
print(img)
```

```
[[[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 ...

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [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 [47]:
```
plt.imshow(img)
```

Out[47]:
```
<matplotlib.image.AxesImage at 0x21496c3c400>
```



In [48]:
```
img.shape
```

Out[48]:
```
(914, 664, 3)
```

In [49]:
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

In [50]:
```
gray.shape
```

Out[50]:
```
(914, 664)
```

In [51]:
```
resized = cv2.resize(gray, (28,28),interpolation = cv2.INTER_AREA)
```

In [52]:
```
resized.shape
```

Out[52]:
```
(28, 28)
```

In [53]:
```
newimg = tf.keras.utils.normalize (resized,axis=1)
```

In [54]:
```
newimg=np.array(newimg).reshape(-1,IMG_SIZE, IMG_SIZE, 1)
```

In [55]:
```
newimg.shape
```

Out[55]:
```
(1, 28, 28, 1)
```

In [56]:
```
predicions = model.predict(newimg)
```

```
1/1 [==============================] - 14s 14s/step
```

In [57]:
```
print (np.argmax(predicions))
```

```
7
```

In [ ]: