```
[2]: import tensorflow as tf
     from tensorflow import keras
     import matplotlib.pyplot as plt
     import random
```

```
[3]: mnist = tf.keras.datasets.mnist
     (x_train, y_train), (x_test,y_test) = mnist.load_data()
```

```
[4]: x_train[0]
     # y_test[0]
     # len(x_test[0][0])
```

```
[4]: 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,   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,  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,
```

```
       [  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]], dtype=uint8)
```

[91]:
```python
x_train = x_train / 255
x_test = x_test / 255
```

[92]:
```python
x_train[0][0]
```

[92]:
```
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.])
```

[93]:
```python
model = keras.Sequential([
    keras.layers.Flatten(input_shape = (28,28)),
    keras.layers.Dense(128,activation = "relu"),
    keras.layers.Dense(10,activation = "softmax")

])
```

[94]:
```python
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 784)	0
dense_4 (Dense)	(None, 128)	100,480
dense_5 (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)

[95]:
```python
model.compile(optimizer = "sgd",
              loss = "sparse_categorical_crossentropy",
              metrics = ["accuracy"])
```

[109]:
```python
history = model.fit(x_train,y_train,validation_data = (x_test,y_test),epochs = 20)
```

```
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9569 - loss: 0.1584 - val_accuracy: 0.9550 - val_loss: 0.1560
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9586 - loss: 0.1497 - val_accuracy: 0.9560 - val_loss: 0.1490
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9601 - loss: 0.1442 - val_accuracy: 0.9575 - val_loss: 0.1422
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9629 - loss: 0.1349 - val_accuracy: 0.9607 - val_loss: 0.1371
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9638 - loss: 0.1338 - val_accuracy: 0.9616 - val_loss: 0.1319
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9646 - loss: 0.1270 - val_accuracy: 0.9626 - val_loss: 0.1270
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9673 - loss: 0.1202 - val_accuracy: 0.9652 - val_loss: 0.1237
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9684 - loss: 0.1159 - val_accuracy: 0.9652 - val_loss: 0.1211
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9692 - loss: 0.1139 - val_accuracy: 0.9667 - val_loss: 0.1162
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9715 - loss: 0.1057 - val_accuracy: 0.9672 - val_loss: 0.1146
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9725 - loss: 0.1015 - val_accuracy: 0.9678 - val_loss: 0.1108
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9725 - loss: 0.0997 - val_accuracy: 0.9689 - val_loss: 0.1072
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9743 - loss: 0.0960 - val_accuracy: 0.9696 - val_loss: 0.1054
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9737 - loss: 0.0954 - val_accuracy: 0.9705 - val_loss: 0.1040
Epoch 15/20
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9762 - loss: 0.0903 - val_accuracy: 0.9704 - val_loss: 0.1022
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9756 - loss: 0.0889 - val_accuracy: 0.9705 - val_loss: 0.0989
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9761 - loss: 0.0875 - val_accuracy: 0.9711 - val_loss: 0.0995
Epoch 18/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9783 - loss: 0.0814 - val_accuracy: 0.9717 - val_loss: 0.0964
Epoch 19/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9794 - loss: 0.0767 - val_accuracy: 0.9716 - val_loss: 0.0949
Epoch 20/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9795 - loss: 0.0781 - val_accuracy: 0.9732 - val_loss: 0.0937
```

[110]:
```
test_loss , test_acc = model.evaluate(x_test,y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9681 - loss: 0.1108
```

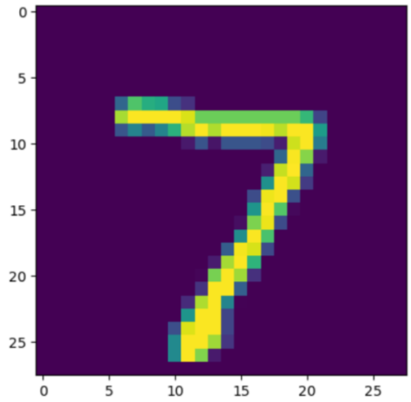
[111]:
```
print("Loss=%.3f"%test_loss)
```

```
Loss=0.094
```

[112]:
```
print("Accuracy=%.3f"%test_acc)
```

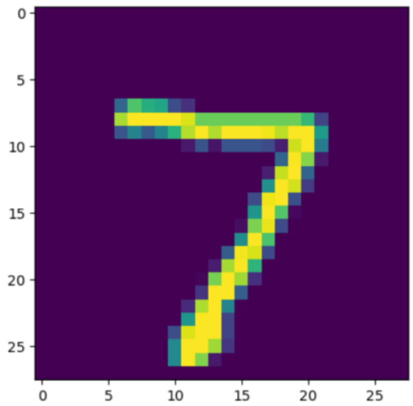
```
Accuracy=0.973
```

[113]:
```
n = random.randint(0,9999)
n = 0
plt.imshow(x_test[n])
plt.show()
```



[114]:
```
predicted_value = model.predict(x_test)
plt.imshow(x_test[n])
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



[115]:
```
# predicted_value
history.history
```

[115]:

```
{'accuracy': [0.9555166959762573,
  0.9580000042915344,
  0.9601666927337646,
  0.9623666405677795,
  0.9645000100135803,
  0.9655166864395142,
  0.9670833349227905,
  0.9681500196456909,
  0.9698666930198669,
  0.9708333611488342,
  0.9717666506767273,
  0.9725333452224731,
  0.973466694355011,
  0.9742000102996826,
  0.9756666421890259,
  0.9759833216667175,
  0.9768833518028259,
  0.9777666926383972,
  0.9779000282287598,
  0.9789999723434448],
 'loss': [0.15915703773498535,
  0.1510932594537735,
  0.14346249401569366,
  0.13699840009212494,
  0.13083772361278534,
  0.12537965178489685,
  0.12020920217037201,
  0.12020920217037201,
  0.11543109267950058,
  0.11101405322551727,
  0.10699454694986343,
  0.10324109345674515,
  0.09974426031112671,
  0.09645798802375793,
  0.09341739118099213,
  0.09047608077526093,
  0.08770239353179932,
  0.08506958931684494,
  0.08266200125217438,
  0.08036860823631287,
  0.0780303105711937],
 'val_accuracy': [0.9549999833106995,
  0.9559999704360962,
  0.9574999809265137,
  0.9606999754905701,
  0.9616000056266785,
  0.9625999927520752,
  0.9652000069618225,
  0.9652000069618225,
  0.96670001745224,
  0.967199981212616,
  0.9678000211715698,
  0.9689000248908997,
  0.9696000218391418,
  0.9704999923706055,
  0.9703999757766724,
  0.9703999757766724,
  0.9704999923706055,
  0.9710999727249146,
  0.9717000126838684,
  0.9715999960899353,
  0.9732000231742859],
 'val_loss': [0.15603896975517273,
  0.14900389313697815,
  0.1422204077243805,
  0.137078195810318,
  0.13194707036018372,
  0.12701231241226196,
  0.12366113066673279,
  0.12112099677324295,
  0.1162414699792862,
  0.11464837938547134,
  0.11082331836223602,
  0.10716228187084198,
  0.1053575798869133,
  0.10402681678533554,
  0.1022346168756485,
  0.09894026815891266,
  0.09946028143167496,
  0.09642958641052246,
  0.09492157399654388,
  0.09367364645004272]}
```

[116]:

```python
print('Predicted Value:',predicted_value[n])
```

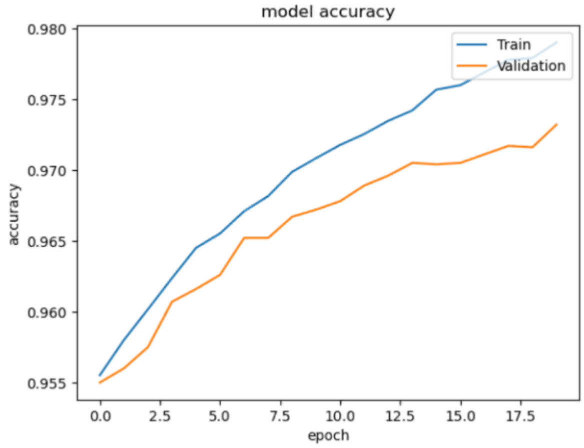
[116]:

```python
print('Predicted Value:',predicted_value[n])
```

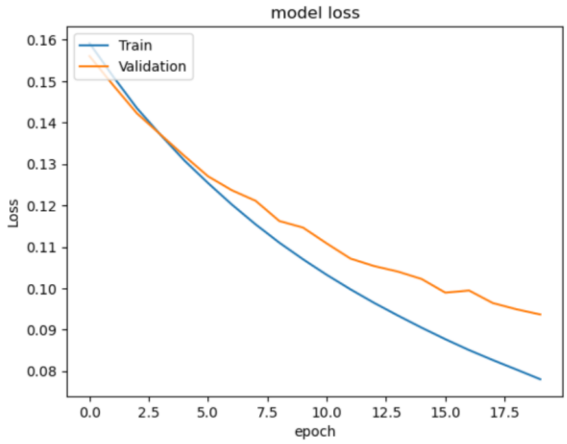

```
Predicted Value: [4.9714945e-06 9.0947175e-08 9.7314522e-05 9.0016244e-04 4.7160459e-08
 4.3946628e-05 9.0947382e-11 9.9890089e-01 1.2419226e-05 4.0144929e-05]
```

[117]:

```python
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 right')
plt.show()
```

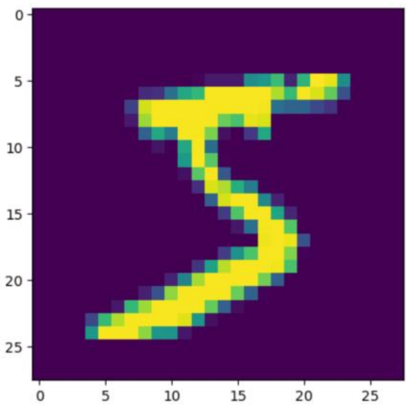


[118]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['Train','Validation'],loc = 'upper left')
plt.show()
```



[106]:
```
plt.imshow(x_train[0])
```

[106]: <matplotlib.image.AxesImage at 0x24c835da480>



[108]:
```
y_train[0]
```

[108]: 5