In [2]:
```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import random
import pandas as pd
import numpy as np
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D
from tensorflow.keras.optimizers import Adam

# Load the CSV files
mnist_train = pd.read_csv('mnist_train.csv')
mnist_test = pd.read_csv('mnist_test.csv')

# Separate features (pixels) and labels
x_train = mnist_train.iloc[:, 1:].values  # All rows, all columns except th
y_train = mnist_train.iloc[:, 0].values   # First column only (labels)
x_test = mnist_test.iloc[:, 1:].values
y_test = mnist_test.iloc[:, 0].values

# Normalize the features (pixels)
x_train = x_train/255.0
x_test = x_test/ 255.0
```

In [8]:
```
model=keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
    keras.layers.Dense(128,activation="relu"),
    keras.layers.Dense(10,activation="softmax")
])
```

```
C:\Users\sd616\anaconda\lib\site-packages\keras\src\layers\reshaping\flatt
en.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument t
o a layer. When using Sequential models, prefer using an `Input(shape)` ob
ject as the first layer in the model instead.
  super().__init__(**kwargs)
```

In [9]:
```
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)

In [19]:
```
model.compile(optimizer="sgd",
loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

In [20]:
```
# Reshape the data to 28x28 pixels and add a channel dimension (1 for grays
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Check if the reshaping is correct
print(x_train.shape)  # Should print (num_samples, 28, 28, 1)
print(x_test.shape)   # Should print (num_samples, 28, 28, 1)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

In [21]:
```
history=model.fit(x_train,
y_train,validation_data=(x_test,y_test),epochs=10)
```

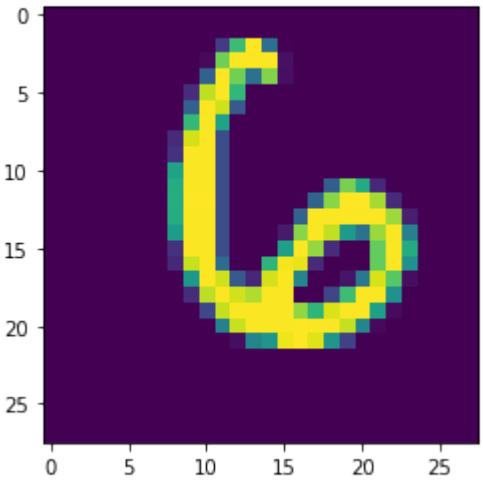
```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.7324 - loss: 1.03
38 - val_accuracy: 0.9008 - val_loss: 0.3590
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9025 - loss: 0.35
27 - val_accuracy: 0.9190 - val_loss: 0.2913
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9164 - loss: 0.29
56 - val_accuracy: 0.9275 - val_loss: 0.2604
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9256 - loss: 0.26
51 - val_accuracy: 0.9336 - val_loss: 0.2357
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9335 - loss: 0.23
66 - val_accuracy: 0.9375 - val_loss: 0.2182
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9393 - loss: 0.21
84 - val_accuracy: 0.9431 - val_loss: 0.2011
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9433 - loss: 0.20
22 - val_accuracy: 0.9445 - val_loss: 0.1892
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9464 - loss: 0.18
81 - val_accuracy: 0.9482 - val_loss: 0.1801
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9518 - loss: 0.17
17 - val_accuracy: 0.9504 - val_loss: 0.1678
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9545 - loss: 0.16
24 - val_accuracy: 0.9519 - val_loss: 0.1615
```

In [23]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%3f" %test_loss)
print("Accuracy=%3f" %test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 942us/step - accuracy: 0.9448 - loss: 0.18
58
Loss=0.161464
Accuracy=0.951900
```

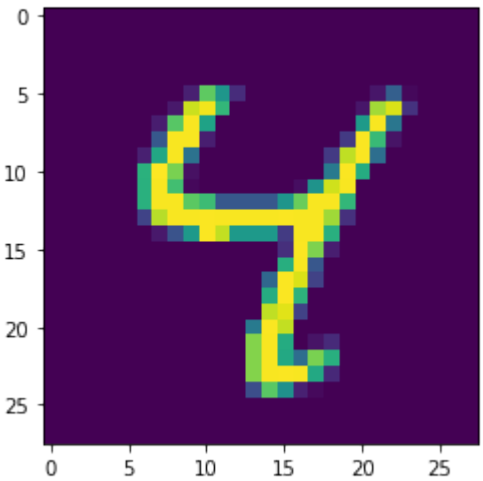
In [34]:
```
n=random.randint(0,9999)
plt.imshow(x_test[88])
```

Out[34]:
```
<matplotlib.image.AxesImage at 0x1f30c487520>
```



In [27]:
```
plt.show()
predicted_value=model.predict(x_test)
plt.imshow(x_test[6])
plt.show()
```

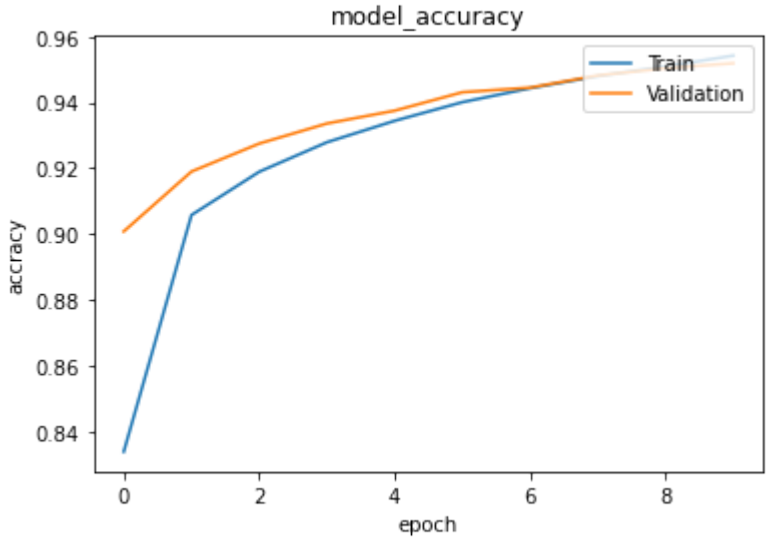
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 826us/step
```



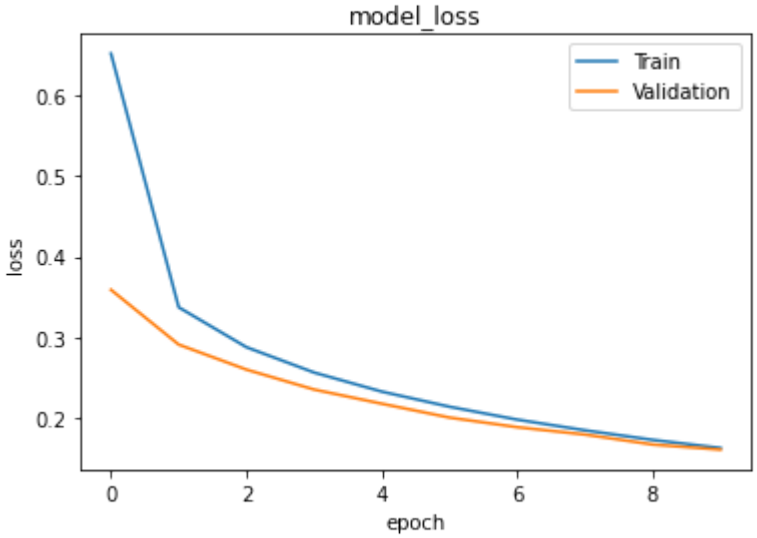
In [30]:
```
print('Predicted value:',predicted_value[n])
```

```
Predicted value: [2.9946582e-06 9.7755164e-01 4.9667986e-04 1.0547031e-02
2.4181115e-04
 2.7306180e-03 3.3848226e-04 1.2099214e-04 7.5388700e-03 4.3090849e-04]
```

In [31]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model_accuracy')
plt.ylabel('accracy')
plt.xlabel('epoch')
plt.legend(['Train','Validation'],loc='upper right')
plt.show()
```



In [32]:
```
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 right')
plt.show()
```



In [ ]: