In [1]:

```
#import all libraries
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import random
```

```
2024-07-30 09:08:01.255121: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, GPU
will not be used.
2024-07-30 09:08:03.038791: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, GPU
will not be used.
2024-07-30 09:08:03.051610: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized t
o use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler
flags.
2024-07-30 09:08:06.944763: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find Ten
sorRT
```

In [2]:

```
#load training and testing data
mnist=tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test)=mnist.load_data()

x_train=x_train / 255
x_test=x_test / 255
```

In [3]:

```
#define model using keras
model=keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
    keras.layers.Dense(128,activation="relu"),
    keras.layers.Dense(10,activation="softmax")
])
```

In [4]:

```
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 [6]:

```
#training model using sgd
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 [==============================] - 2s 829us/step - loss: 0.2577 - accuracy: 0.9283 - val_loss: 0.2374 - val_
accuracy: 0.9321
Epoch 2/10
1875/1875 [==============================] - 2s 806us/step - loss: 0.2346 - accuracy: 0.9353 - val_loss: 0.2168 - val_
accuracy: 0.9383
Epoch 3/10
1875/1875 [==============================] - 2s 813us/step - loss: 0.2161 - accuracy: 0.9402 - val_loss: 0.2030 - val_
accuracy: 0.9409
Epoch 4/10
1875/1875 [==============================] - 1s 796us/step - loss: 0.2001 - accuracy: 0.9445 - val_loss: 0.1909 - val_
accuracy: 0.9447
Epoch 5/10
1875/1875 [==============================] - 1s 794us/step - loss: 0.1870 - accuracy: 0.9483 - val_loss: 0.1770 - val_
accuracy: 0.9477
Epoch 6/10
1875/1875 [==============================] - 1s 796us/step - loss: 0.1752 - accuracy: 0.9511 - val_loss: 0.1684 - val_
accuracy: 0.9488
Epoch 7/10
1875/1875 [==============================] - 2s 807us/step - loss: 0.1648 - accuracy: 0.9534 - val_loss: 0.1596 - val_
accuracy: 0.9533
Epoch 8/10
1875/1875 [==============================] - 1s 793us/step - loss: 0.1557 - accuracy: 0.9561 - val_loss: 0.1544 - val_
accuracy: 0.9540
Epoch 9/10
1875/1875 [==============================] - 2s 806us/step - loss: 0.1474 - accuracy: 0.9587 - val_loss: 0.1464 - val_
accuracy: 0.9579
Epoch 10/10
1875/1875 [==============================] - 2s 817us/step - loss: 0.1398 - accuracy: 0.9608 - val_loss: 0.1393 - val_
accuracy: 0.9602
```

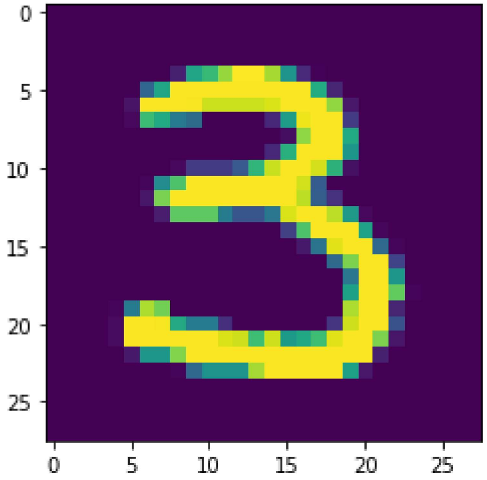
In [9]:

```
#Evaluate model
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%3f" %test_loss)
print("Accuracy=%3f" %test_acc)
```

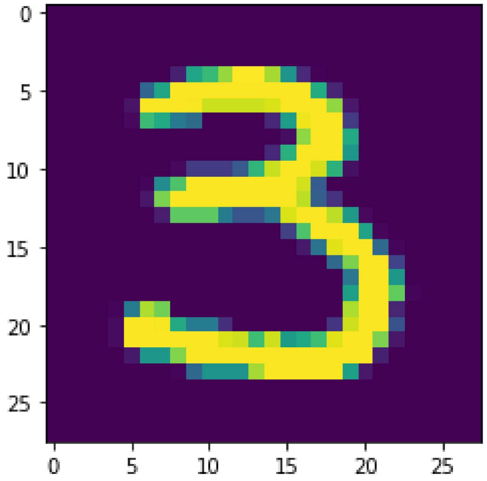
```
313/313 [==============================] - 0s 585us/step - loss: 0.1393 - accuracy: 0.9602
Loss=0.139341
Accuracy=0.960200
```

In [16]:

```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()
```



```
313/313 [==============================] - 0s 496us/step
```

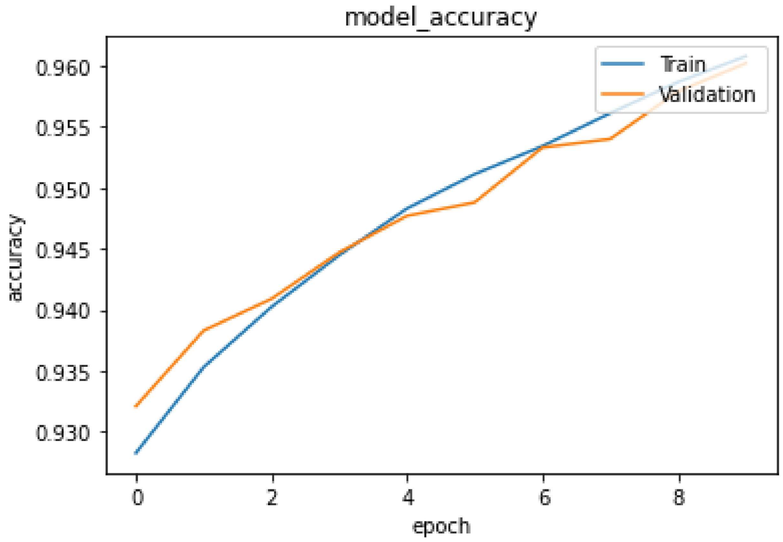


```
In [12]: print('Predicted value:', predicted_value[n])
```

```
Predicted value: [3.5806042e-06 1.1032747e-04 3.1124946e-04 1.1362509e-02 1.4568138e-03
 3.8088212e-04 3.6567468e-07 5.4844362e-03 1.7520562e-03 9.7913784e-01]
```

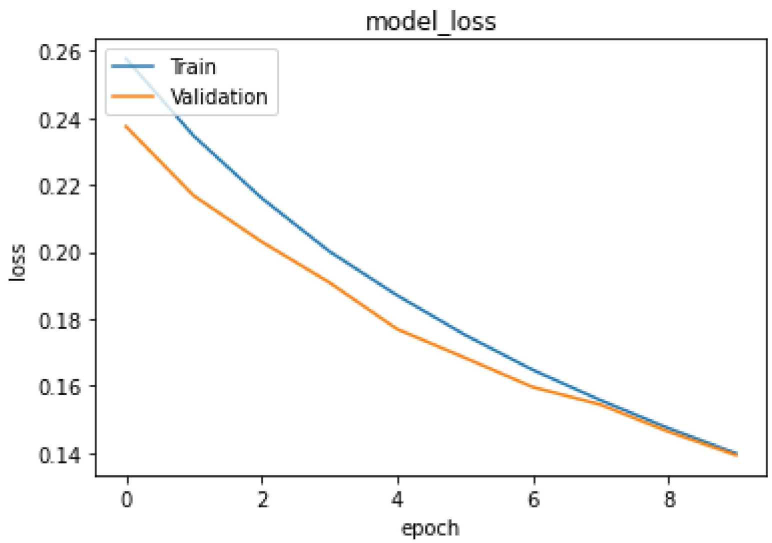
In [13]:

```
#training 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 right')
plt.show()
```



In [15]:

```
#training loss
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()
```


model_loss
Train
Validation
loss
0.26
0.24
0.22
0.20
0.18
0.16
0.14
epoch
0
2
4
6
8

In [ ]: