In [28]:

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

In [29]:

```
# b. Load the training and testing data (MNIST/CIFAR10)
mnist = tf.keras.datasets.mnist
```

In [30]:

```

# Splitting it into traning and testing data
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [31]:

```

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

In [32]:

```
# c. Define the network architecture using Keras
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_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_2 (Flatten)         (None, 784)               0         
                                                                 
 dense_4 (Dense)             (None, 128)               100480    
                                                                 
 dense_5 (Dense)             (None, 10)                1290      
                                                                 
=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [36]:

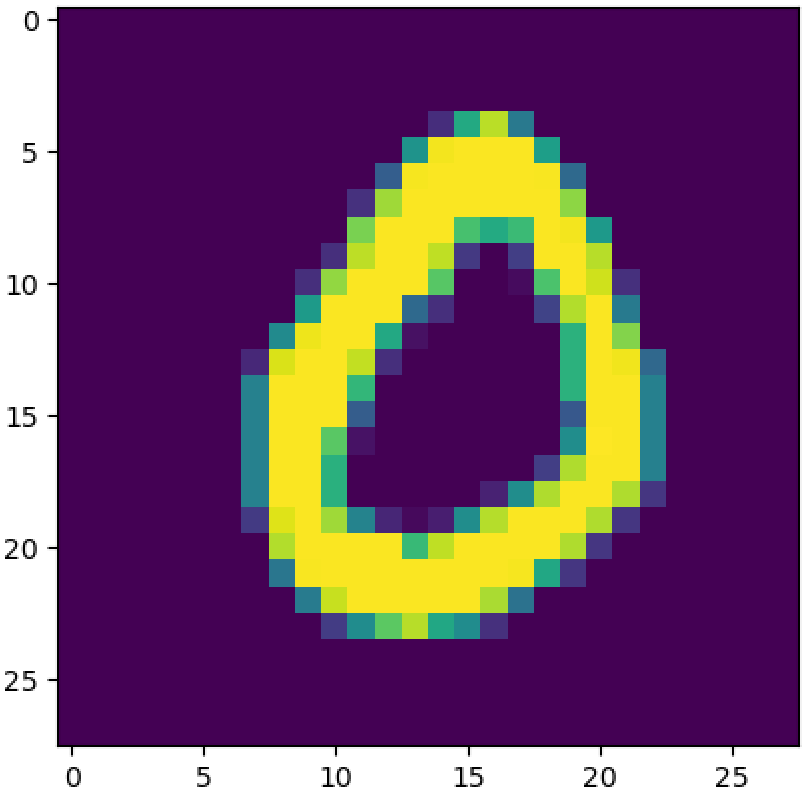
```
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train,
                    validation_data=(x_test, y_test),
                    epochs=10,
                    verbose=1)
```

```
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1609 -
accuracy: 0.9553 - val_loss: 0.1607 - val_accuracy: 0.9534
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1525 -
accuracy: 0.9575 - val_loss: 0.1539 - val_accuracy: 0.9558
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1448 -
accuracy: 0.9599 - val_loss: 0.1458 - val_accuracy: 0.9571
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1380 -
accuracy: 0.9614 - val_loss: 0.1389 - val_accuracy: 0.9592
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1316 -
accuracy: 0.9632 - val_loss: 0.1351 - val_accuracy: 0.9606
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1258 -
accuracy: 0.9653 - val_loss: 0.1312 - val_accuracy: 0.9606
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1205 -
accuracy: 0.9668 - val_loss: 0.1270 - val_accuracy: 0.9626
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1158 -
accuracy: 0.9678 - val_loss: 0.1230 - val_accuracy: 0.9640
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1112 -
accuracy: 0.9690 - val_loss: 0.1196 - val_accuracy: 0.9642
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1071 -
accuracy: 0.9708 - val_loss: 0.1149 - val_accuracy: 0.9656
```

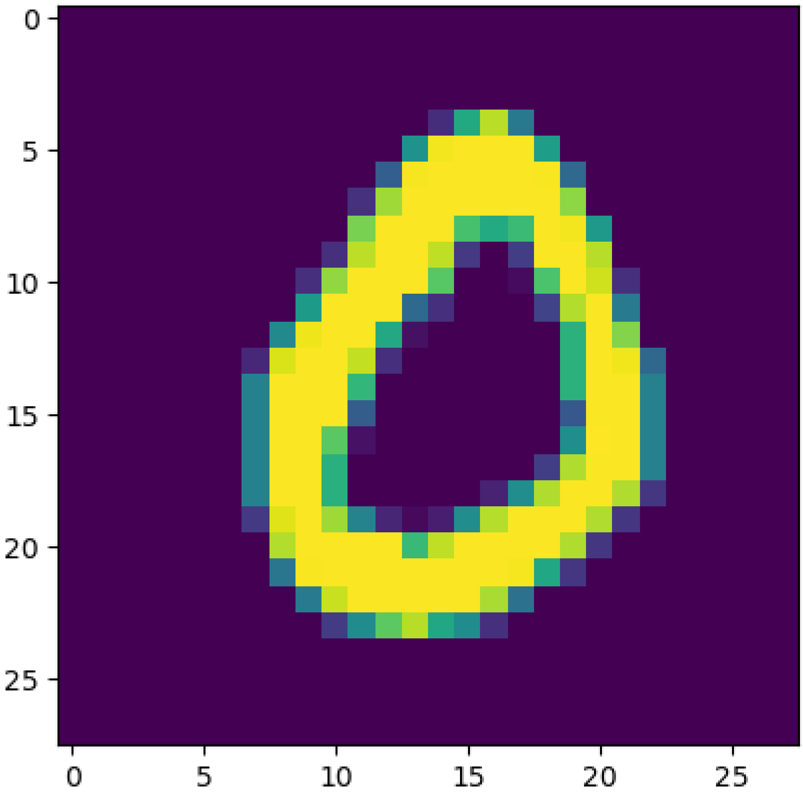
In [37]:

```python

# e. Evaluate the network

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

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()

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

```
313/313 [==============================] - 0s 1ms/step - loss: 0.1149 - ac
curacy: 0.9656
Loss=0.115
Accuracy=0.966
```



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

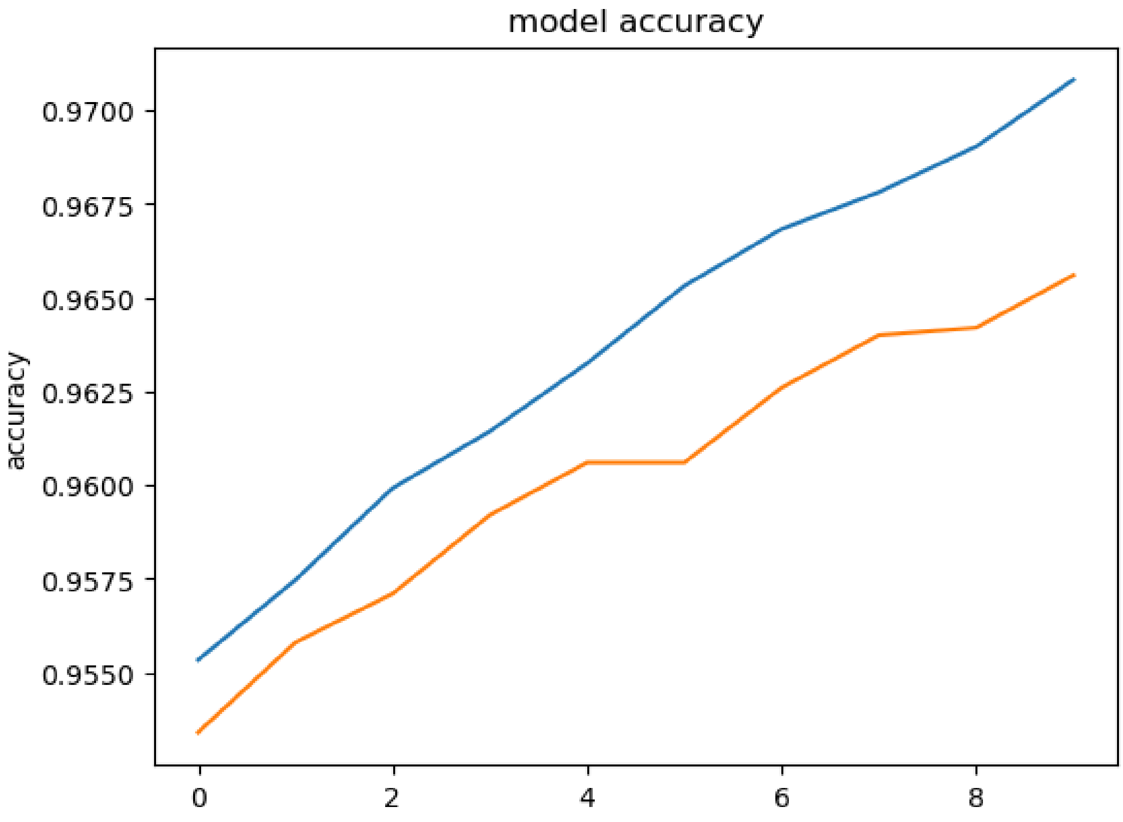


```
Predicted value: [9.9978179e-01 1.5406085e-08 3.6826103e-05 1.1992198e-06
4.3907779e-08
 4.1520416e-06 1.7369386e-04 2.9695937e-07 5.9352038e-07 1.4073751e-06]
```

In [38]:

```
# f. Plot the training loss and accuracy

 # plotting the training Accuracy

plt.plot(history.history['accuracy'])
# plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlable('epoch')
plt.legend(['Train','Validation'],loc='upper left')
plt.show()
```

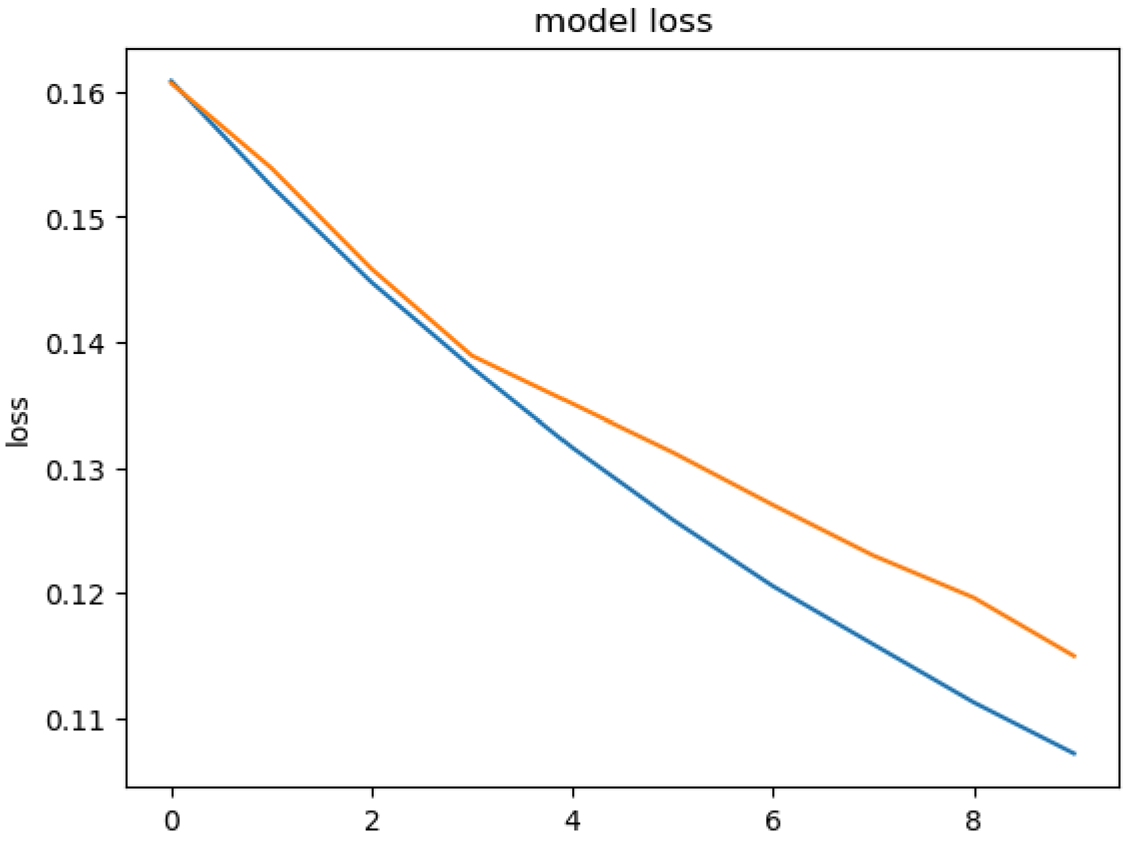
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call las
t)
Cell In[38], line 10
      8 plt.title('model accuracy')
      9 plt.ylabel('accuracy')
---> 10 plt.xlable('epoch')
     11 plt.legend(['Train','Validation'],loc='upper left')
     12 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'xlable'
```



In [39]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlable('epoch')
plt.legend(['Train','Validation'],loc='upper right')
plt.show()
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
Cell In[39], line 5
      3 plt.title('model loss')
      4 plt.ylabel('loss')
----> 5 plt.xlable('epoch')
      6 plt.legend(['Train','Validation'],loc='upper right')
      7 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'xlable'
```



In [ ]: