In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import random
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, Dense, MaxPooling2D, Flatten
```

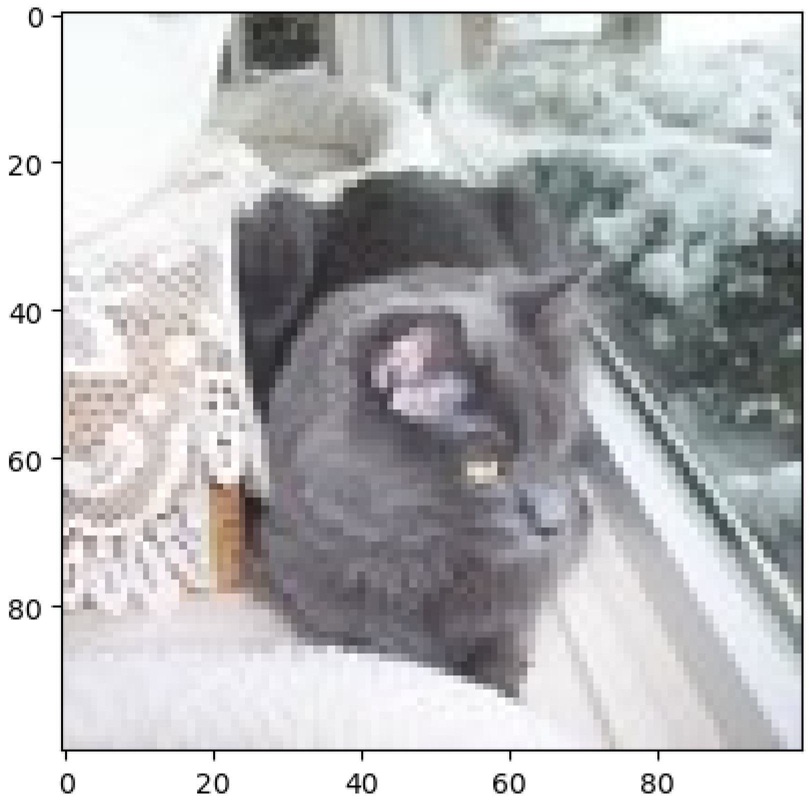
In [2]:
```python
X_train = np.loadtxt('input.csv', delimiter = ',')
Y_train = np.loadtxt('labels.csv', delimiter = ',')

x_test = np.loadtxt('input_test.csv', delimiter=',')
y_test = np.loadtxt('labels_test.csv', delimiter = ',')
```

In [3]:
```python
X_train = X_train.reshape(len(X_train), 100, 100, 3)
Y_train = Y_train.reshape(len(Y_train), 1)

x_test = x_test.reshape(len(x_test), 100, 100, 3)
y_test = y_test.reshape(len(y_test), 1)

X_train = X_train/255.0
x_test = x_test/255.0
```

In [4]:
```python
print('Shape of X_train: ',X_train.shape)
print('Shape of Y_train: ',Y_train.shape)
print('Shape of x_test: ',x_test.shape)
print('Shape of y_test: ',y_test.shape)
```

```
Shape of X_train:  (2000, 100, 100, 3)
Shape of Y_train:  (2000, 1)
Shape of x_test:  (400, 100, 100, 3)
Shape of y_test:  (400, 1)
```

In [5]:
```python
idx = random.randint(0, len(X_train))
plt.imshow(X_train[idx, :])
plt.show()
```



In [6]:
```
model = Sequential([
    Conv2D(32, (3,3), activation = 'relu', input_shape=(100,100,3)),
    MaxPooling2D((2,2)),

    Conv2D(32, (3,3), activation = 'relu'),
    MaxPooling2D((2,2)),

    Flatten(),
    Dense(64, activation = 'relu'),
    Dense(1, activation = 'sigmoid'),
])
```

```
C:\Users\rahul\miniconda3\Lib\site-packages\keras\src\layers\convolutional\bas
e_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument t
o a layer. When using Sequential models, prefer using an `Input(shape)` object
as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [7]:
```
model = Sequential()

model.add(Conv2D(32, (3,3), activation = 'relu', input_shape = (100, 100, 3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32, (3,3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(64, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

```
In [7]: model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy
```

```
In [8]: model.fit(X_train, Y_train, epochs = 10, batch_size = 64)
```

```
Epoch 1/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 189ms/step - accuracy: 0.5065 - loss: 0.7329
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 186ms/step - accuracy: 0.5665 - loss: 0.6743
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 175ms/step - accuracy: 0.6855 - loss: 0.6059
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 186ms/step - accuracy: 0.6985 - loss: 0.5667
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 185ms/step - accuracy: 0.7530 - loss: 0.5025
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 190ms/step - accuracy: 0.7985 - loss: 0.4500
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 181ms/step - accuracy: 0.8360 - loss: 0.3819
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 189ms/step - accuracy: 0.8455 - loss: 0.3565
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 189ms/step - accuracy: 0.8845 - loss: 0.2884
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 186ms/step - accuracy: 0.9145 - loss: 0.2322
```

```
Out[8]: <keras.src.callbacks.history.History at 0x1b441adbf50>
```

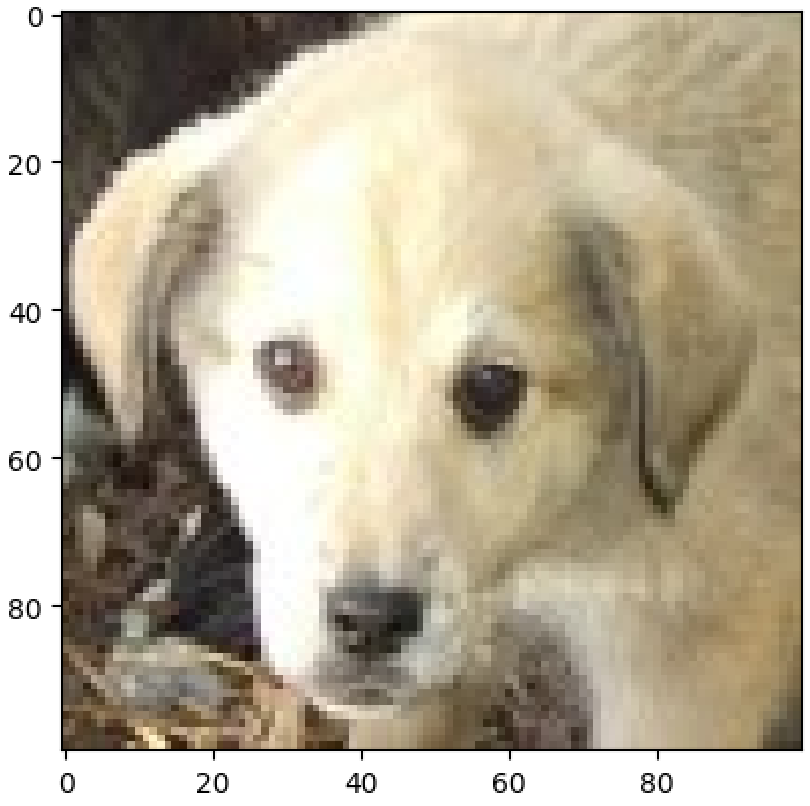
```
In [10]: model.evaluate(x_test, y_test)
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.7200 - loss: 0.7369
```

```
Out[10]: [0.7368872761726379, 0.7200000286102295]
```

```
In [11]: idx2 = random.randint(0, len(x_test))
         plt.imshow(x_test[idx2, :])

         y_pred = model.predict(x_test[idx2, :].reshape(1, 100, 100, 3))
         y_pred = y_pred > 0.5

         if(y_pred == 0):
             pred = 'dog'
         else:
             pred = 'cat'

         print('Prediction: ', pred)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step
Prediction:  dog
```



```
In [ ]:
```

```
In [ ]:
```