In [3]:
```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
cp: cannot stat 'kaggle.json': No such file or directory
```

In [4]:
```
!kaggle datasets download -d salader/dogs-vs-cats
```

```
Dataset URL: https://www.kaggle.com/datasets/salader/dogs-vs-cats
License(s): unknown
Downloading dogs-vs-cats.zip to /content
100% 1.06G/1.06G [01:03<00:00, 18.6MB/s]
100% 1.06G/1.06G [01:03<00:00, 18.0MB/s]
```

In [5]:
```
from zipfile import ZipFile
```

In [6]:
```
zip_ref=ZipFile('/content/dogs-vs-cats.zip','r')
zip_ref.extractall('/content/')
zip_ref.close()
```

In [7]:
```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Conv2D,MaxPool2D,Dense,BatchNormalization,Dropou
t,Flatten
```

In [8]:
```
train_ds=keras.utils.image_dataset_from_directory(directory='/content/trai
n',labels='inferred',label_mode='int',batch_size=32,image_size=(256,256))

validation_ds=keras.utils.image_dataset_from_directory(directory='/content/
test',labels='inferred',label_mode='int',batch_size=32,image_size=(256,25
6))
```

```
Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.
```

In [9]:
```
train_ds
```

Out[9]:
```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype
=tf.float32, name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=Non
e))>
```

In [10]:
```
def process(img,label):
  img=tf.cast(img/255.0, tf.float32)
  return img,label

train_ds=train_ds.map(process)
```

In [11]:
```
validation_ds=validation_ds.map(process)
```

In [12]:
```
model=Sequential()
model.add(Conv2D(filters=32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2,2),strides=2,padding='valid'))


model.add(Conv2D(filters=64,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2,2),strides=2,padding='valid'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to 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 [13]:
```
model.add(Conv2D(filters=128,kernel_size=(3,3),padding='valid',activation
='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2,2),strides=2,padding='valid'))


model.add(Conv2D(filters=256,kernel_size=(3,3),padding='valid',activation
='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	
conv2d (Conv2D)	(None, 254, 254, 32)	
batch_normalization (BatchNormalization)	(None, 254, 254, 32)	
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	
conv2d_1 (Conv2D)	(None, 125, 125, 64)	
batch_normalization_1 (BatchNormalization)	(None, 125, 125, 64)	
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	
conv2d_2 (Conv2D)	(None, 60, 60, 128)	
batch_normalization_2 (BatchNormalization)	(None, 60, 60, 128)	
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	
conv2d_3 (Conv2D)	(None, 28, 28, 256)	
batch_normalization_3 (BatchNormalization)	(None, 28, 28, 256)	
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	
flatten (Flatten)	(None, 50176)	
dense (Dense)	(None, 128)	
dropout (Dropout)	(None, 128)	
dense_1 (Dense)	(None, 64)	
dropout_1 (Dropout)	(None, 64)	
dense_2 (Dense)	(None, 1)	

Total params: 6,821,313 (26.02 MB)

Trainable params: 6,820,353 (26.02 MB)

Non-trainable params: 960 (3.75 KB)

In [14]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [15]:
```
history=model.fit(train_ds,batch_size=32,epochs=10,validation_data=validation_ds)
```

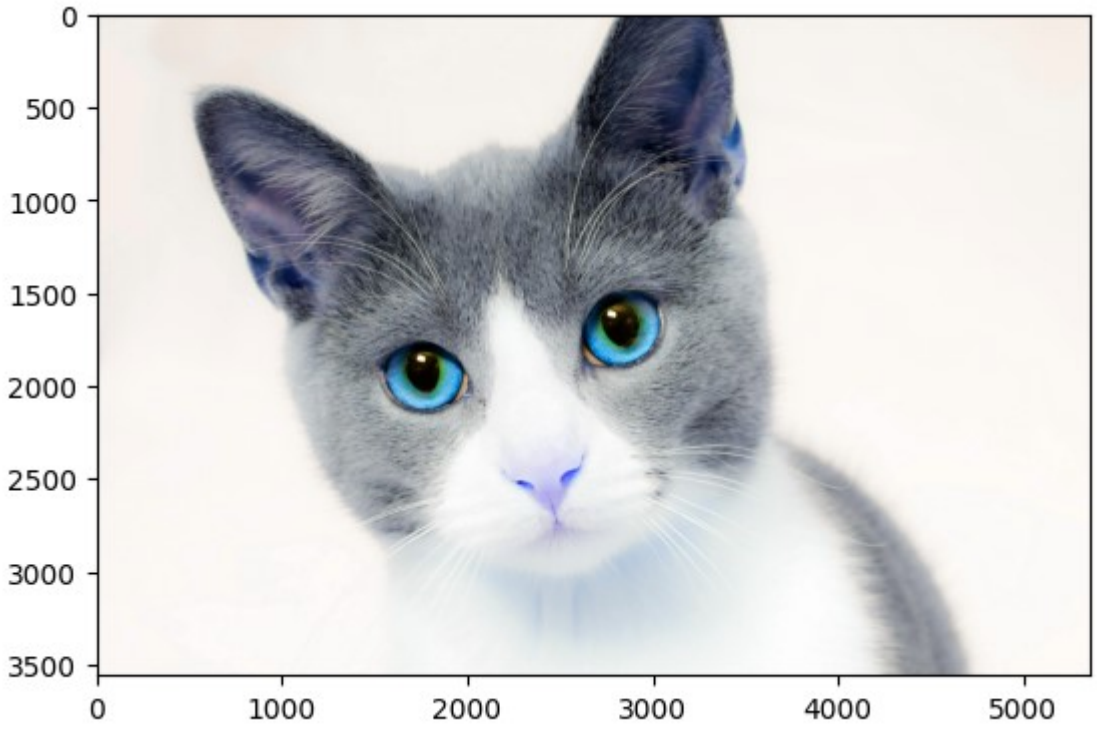
```
Epoch 1/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 69s 88ms/step - accuracy: 0.5601 - loss: 1.7456 - val_accuracy: 0.5782 - val_loss: 0.6563
Epoch 2/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 53s 85ms/step - accuracy: 0.6633 - loss: 0.6142 - val_accuracy: 0.6890 - val_loss: 0.5793
Epoch 3/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 83s 87ms/step - accuracy: 0.7317 - loss: 0.5249 - val_accuracy: 0.7222 - val_loss: 0.5587
Epoch 4/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 81s 85ms/step - accuracy: 0.7976 - loss: 0.4405 - val_accuracy: 0.5350 - val_loss: 0.7920
Epoch 5/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 83s 87ms/step - accuracy: 0.8387 - loss: 0.3599 - val_accuracy: 0.7402 - val_loss: 0.5389
Epoch 6/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 81s 85ms/step - accuracy: 0.8805 - loss: 0.2820 - val_accuracy: 0.8524 - val_loss: 0.3479
Epoch 7/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 91ms/step - accuracy: 0.9184 - loss: 0.2145 - val_accuracy: 0.8888 - val_loss: 0.2781
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 91ms/step - accuracy: 0.9383 - loss: 0.1540 - val_accuracy: 0.8322 - val_loss: 0.4849
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 91ms/step - accuracy: 0.9564 - loss: 0.1156 - val_accuracy: 0.8796 - val_loss: 0.3672
Epoch 10/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 91ms/step - accuracy: 0.9664 - loss: 0.0912 - val_accuracy: 0.8864 - val_loss: 0.4066
```

In [16]:
```
import cv2
```

In [18]:
```
import matplotlib.pyplot as plt
```

In [19]:
```
new_img=cv2.imread('/content/predict_img.jpg')
plt.imshow(new_img)
```

Out[19]:
```
<matplotlib.image.AxesImage at 0x7c80b9230790>
```



In [20]:
```
new_img.shape
```

Out[20]:
```
(3560, 5360, 3)
```

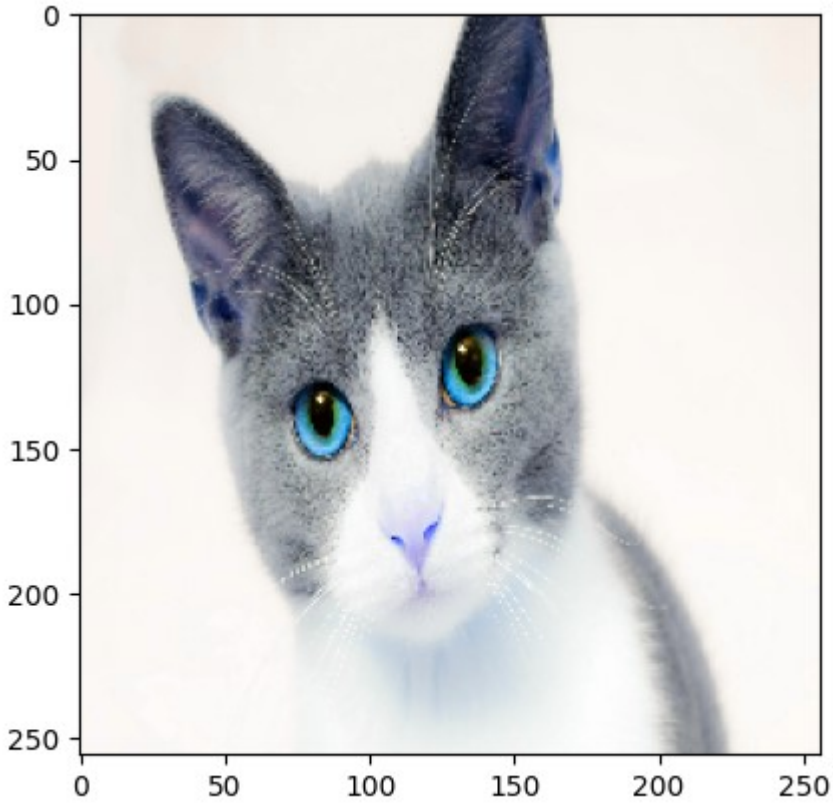
In [23]:
```
new_img.size
```

Out[23]:
```
57244800
```

In [24]:
```
new_img=cv2.resize(new_img,(256,256))
```

In [25]:
```
plt.imshow(new_img)
```

Out[25]:
```
<matplotlib.image.AxesImage at 0x7c80b915d4b0>
```



In [26]:
```
new_img=new_img.reshape((1,256,256,3))
```

In [27]:
```
model.predict(new_img)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 967ms/step
```

Out[27]:
```
array([[0.]], dtype=float32)
```

In [35]:
```
new_img1=cv2.imread('/content/new_img1.jpg')
```

In [36]:
```
plt.imshow(new_img1)
```

Out[36]:
```
<matplotlib.image.AxesImage at 0x7c80c1fae560>
```

In [37]:
```
new_img1=cv2.resize(new_img1,(256,256))
```

In [38]:
```
new_img1=new_img1.reshape(1,256,256,3)
```

In [39]:
```
model.predict(new_img1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```

Out[39]:
```
array([[1.]], dtype=float32)
```

In [ ]:
```

```