Cat vs Dog Classifier

(using Trasfer Learning)

> Transfer learning is a machine learning technique where a model trained on one task or dataset is used as a starting point for training a model on a different, but related, task or dataset. Instead of training a model from scratch, transfer learning leverages the knowledge and features learned from the source task or dataset to improve the learning process for the target task.

Install Kaggle Library

In [1]:
```
!pip install Kaggle
```

```
Requirement already satisfied: Kaggle in e:\anaconda\lib\site-packages (1.5.16)
Requirement already satisfied: six>=1.10 in e:\anaconda\lib\site-packages (from Kaggl
e) (1.16.0)
Requirement already satisfied: certifi in e:\anaconda\lib\site-packages (from Kaggle)
(2023.7.22)
Requirement already satisfied: python-dateutil in e:\anaconda\lib\site-packages (from
Kaggle) (2.8.2)
Requirement already satisfied: requests in e:\anaconda\lib\site-packages (from Kaggle)
(2.31.0)
Requirement already satisfied: tqdm in e:\anaconda\lib\site-packages (from Kaggle) (4.
65.0)
Requirement already satisfied: python-slugify in e:\anaconda\lib\site-packages (from K
aggle) (5.0.2)
Requirement already satisfied: urllib3 in e:\anaconda\lib\site-packages (from Kaggle)
(1.26.16)
Requirement already satisfied: bleach in e:\anaconda\lib\site-packages (from Kaggle)
(4.1.0)
Requirement already satisfied: packaging in e:\anaconda\lib\site-packages (from bleach
->Kaggle) (23.0)
Requirement already satisfied: webencodings in e:\anaconda\lib\site-packages (from ble
ach->Kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in e:\anaconda\lib\site-packages (f
rom python-slugify->Kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in e:\anaconda\lib\site-packag
es (from requests->Kaggle) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in e:\anaconda\lib\site-packages (from req
uests->Kaggle) (3.4)
Requirement already satisfied: colorama in e:\anaconda\lib\site-packages (from tqdm->K
aggle) (0.4.6)
```

Configure the path to Kaggle.json file

In [11]:
```
# !mkdir -p ~/.kaggle
# !cp kaggle.json ~/.kaggle/
# !chmod 600 ~/.kaggle/kaggle.json

import os

# Create the .kaggle directory
kaggle_dir = os.path.expanduser("~/.kaggle")
os.makedirs(kaggle_dir, exist_ok=True)

# Copy the kaggle.json file to the .kaggle directory
api_key_content = '{"username":"you_username","key":"your_APIKEY"}'  # Replace with you
api_key_path = os.path.join(kaggle_dir, "kaggle.json")
with open(api_key_path, "w") as api_key_file:
    api_key_file.write(api_key_content)

# Set appropriate permissions
os.chmod(api_key_path, 0o600)
```

Importing the Dog vs Cat Dataset from Kaggle

In [13]:
```
!kaggle competitions download -c dogs-vs-cats
```

```
 98%|#########8| 797M/812M [04:44<00:04, 3.77MB/s]
 98%|#########8| 798M/812M [04:45<00:03, 3.88MB/s]
 98%|#########8| 799M/812M [04:45<00:03, 3.87MB/s]
 99%|#########8| 800M/812M [04:45<00:03, 3.87MB/s]
 99%|#########8| 801M/812M [04:46<00:03, 3.86MB/s]
 99%|#########8| 802M/812M [04:46<00:02, 3.78MB/s]
 99%|#########8| 803M/812M [04:46<00:02, 3.88MB/s]
 99%|#########8| 804M/812M [04:46<00:02, 3.87MB/s]
 99%|#########9| 805M/812M [04:47<00:01, 3.86MB/s]
 99%|#########9| 806M/812M [04:47<00:01, 3.86MB/s]
 99%|#########9| 807M/812M [04:47<00:01, 3.85MB/s]
 99%|#########9| 808M/812M [04:48<00:01, 3.38MB/s]
100%|#########9| 809M/812M [04:48<00:00, 3.60MB/s]
100%|#########9| 810M/812M [04:48<00:00, 3.77MB/s]
100%|#########9| 811M/812M [04:48<00:00, 3.91MB/s]
100%|#########9| 812M/812M [04:49<00:00, 3.98MB/s]
100%|##########| 812M/812M [04:49<00:00, 2.95MB/s]

Downloading dogs-vs-cats.zip to E:\Source Codes\AIML\CatVsDog Classifier
```

In [2]:
```
# !ls

!dir
```

```
 Volume in drive E has no label.
 Volume Serial Number is E8E5-6307

 Directory of E:\Source Codes\AIML\CatVsDog Classifier

21-08-2023  15:09    <DIR>          .
21-08-2023  11:05    <DIR>          ..
21-08-2023  11:17    <DIR>          .ipynb_checkpoints
21-08-2023  12:59    <DIR>          catsdata
21-08-2023  14:52         3,749,256 CatVsDog Classification - Transfer Learning - Cola
boratory.pdf
21-08-2023  15:08         2,341,425 CatVsDog.pdf
11-12-2019  04:19       851,576,689 dogs-vs-cats.zip
21-08-2023  13:00    <DIR>          dogsdata
21-08-2023  15:09         1,525,769 DogVsCatClassifier.ipynb
21-08-2023  13:04    <DIR>          image resized
21-08-2023  11:05                63 kaggle.json
21-08-2023  11:34            88,903 sampleSubmission.csv
21-08-2023  12:50    <DIR>          test data
21-08-2023  11:34       284,321,224 test1.zip
21-08-2023  13:02    <DIR>          train
21-08-2023  11:34       569,546,721 train.zip
               8 File(s)  1,713,150,050 bytes
               8 Dir(s)  23,871,836,160 bytes free
```

Extracting the Zip file

In [17]:
```
from zipfile import ZipFile

dataset = 'your-main-directory\CatVsDog Classifier/dogs-vs-cats.zip'

with ZipFile(dataset, 'r') as zip:
  zip.extractall()
  print('The dataset is extracted')
```

```
The dataset is extracted
```

Extracting the Compressed Dataset

In [18]:
```
from zipfile import ZipFile

dataset = 'your-main-directory\CatVsDog Classifier/train.zip'

with ZipFile(dataset, 'r') as zip:
  zip.extractall()
  print('The dataset is extracted')
```

```
The dataset is extracted
```

In [3]:
```
import os
path, dirs, files = next(os.walk('your-main-directory\CatVsDog Classifier/train'))
file_count = len(files)
print('Number of images: ', file_count)
```

```
Number of images:  2000
```

Printing the name of the Image files

In [4]:
```
file_names = os.listdir('your-main-directory\CatVsDog Classifier/train/')
print(file_names)
```

```
3.jpg', 'dog.904.jpg', 'dog.905.jpg', 'dog.906.jpg', 'dog.907.jpg', 'dog.908.jpg',
'dog.909.jpg', 'dog.91.jpg', 'dog.910.jpg', 'dog.911.jpg', 'dog.912.jpg', 'dog.913.j
pg', 'dog.914.jpg', 'dog.915.jpg', 'dog.916.jpg', 'dog.917.jpg', 'dog.918.jpg', 'do
g.919.jpg', 'dog.92.jpg', 'dog.920.jpg', 'dog.921.jpg', 'dog.922.jpg', 'dog.923.jp
g', 'dog.924.jpg', 'dog.925.jpg', 'dog.926.jpg', 'dog.927.jpg', 'dog.928.jpg', 'dog.
929.jpg', 'dog.93.jpg', 'dog.930.jpg', 'dog.931.jpg', 'dog.932.jpg', 'dog.933.jpg',
'dog.934.jpg', 'dog.935.jpg', 'dog.936.jpg', 'dog.937.jpg', 'dog.938.jpg', 'dog.939.
jpg', 'dog.94.jpg', 'dog.940.jpg', 'dog.941.jpg', 'dog.942.jpg', 'dog.943.jpg', 'do
g.944.jpg', 'dog.945.jpg', 'dog.946.jpg', 'dog.947.jpg', 'dog.948.jpg', 'dog.949.jp
g', 'dog.95.jpg', 'dog.950.jpg', 'dog.951.jpg', 'dog.952.jpg', 'dog.953.jpg', 'dog.9
54.jpg', 'dog.955.jpg', 'dog.956.jpg', 'dog.957.jpg', 'dog.958.jpg', 'dog.959.jpg',
'dog.96.jpg', 'dog.960.jpg', 'dog.961.jpg', 'dog.962.jpg', 'dog.963.jpg', 'dog.964.j
pg', 'dog.965.jpg', 'dog.966.jpg', 'dog.967.jpg', 'dog.968.jpg', 'dog.969.jpg', 'do
g.97.jpg', 'dog.970.jpg', 'dog.971.jpg', 'dog.972.jpg', 'dog.973.jpg', 'dog.974.jp
g', 'dog.975.jpg', 'dog.976.jpg', 'dog.977.jpg', 'dog.978.jpg', 'dog.979.jpg', 'dog.
98.jpg', 'dog.980.jpg', 'dog.981.jpg', 'dog.982.jpg', 'dog.983.jpg', 'dog.984.jpg',
'dog.985.jpg', 'dog.986.jpg', 'dog.987.jpg', 'dog.988.jpg', 'dog.989.jpg', 'dog.99.j
pg', 'dog.990.jpg', 'dog.991.jpg', 'dog.992.jpg', 'dog.993.jpg', 'dog.994.jpg', 'do
g.995.jpg', 'dog.996.jpg', 'dog.997.jpg', 'dog.998.jpg', 'dog.999.jpg']
```

Important Note: The dataset has a total of 25000 images of cats and dogs. My local machine doesn't have the required power to handle this huge data, and go through it to train the model. So I reduced the dataset to 2000 images, 1000 of cats and dogs respectively, so that I can compute the data, and run the model in my machine in small amout of time with my CPU resources. While the Google Collab environment is cloud based and has it's seperate RAM, it can handle the large dataset properly. If your System is powerful enough, you can use the whole dataset of 25000 images for training and testing.

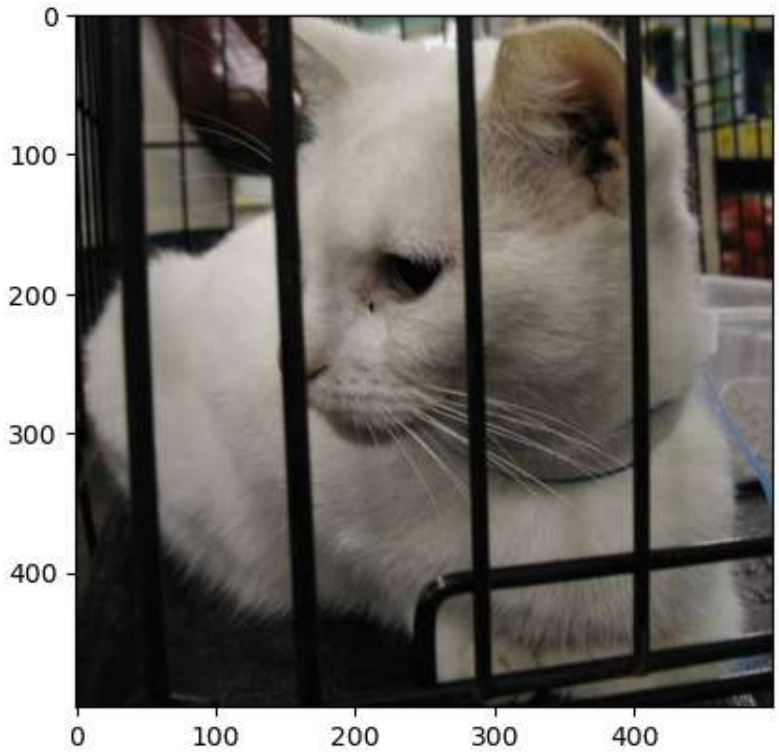
INSTALLING DEPENDENCIES

In [5]:
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
import cv2
```

Displaying example images of a Dog and a Cat

In [6]:
```
# DOG IMAGE
img = mpimg.imread('your-main-directory\CatVsDog Classifier/train/dog.342.jpg')
imgplt = plt.imshow(img)
plt.show()
```

In [7]:
```
# CAT IMAGE
img = mpimg.imread('your-main-directory\CatVsDog Classifier/train/cat.658.jpg')
imgplt = plt.imshow(img)
plt.show()
```



In [8]:
```
file_names = os.listdir('your-main-directory\CatVsDog Classifier/train/')

for i in range(5):

  name = file_names[i]
  print(name[0:3])
```

```
cat
cat
cat
cat
cat
```

Counting the number of Cats and Dogs (image files)

In [9]:
```
file_names = os.listdir('your-main-directory\CatVsDog Classifier/train/')

dog_count = 0
cat_count = 0

for img_file in file_names:

  name = img_file[0:3]

  if name == 'dog':
    dog_count += 1

  else:
    cat_count += 1

print('Number of dog images =', dog_count)
print('Number of cat images =', cat_count)
```

```
Number of dog images = 1000
Number of cat images = 1000
```

RESIZING THE IMAGES

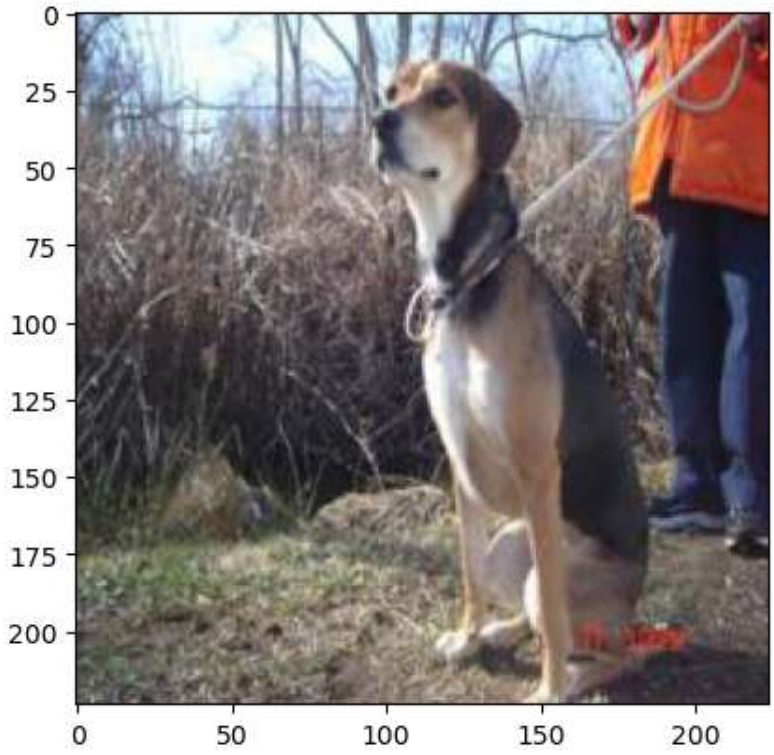
- Resizing all the images to 224x224

Creating a directory for resized images

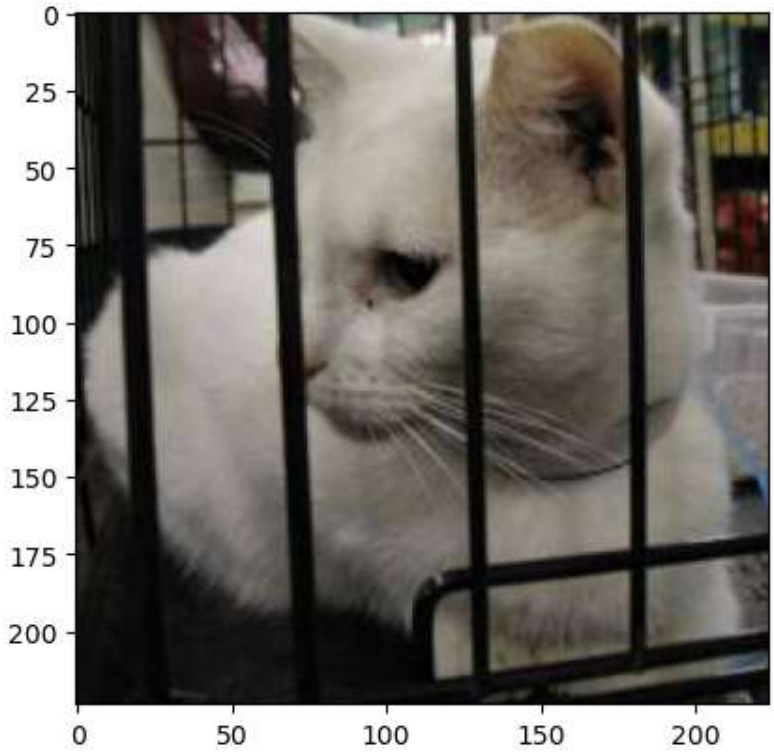
In [70]:
```
os.mkdir('your-main-directory\CatVsDog Classifier/image resized')
```

In [10]:
```
original_folder = 'your-main-directory\CatVsDog Classifier/train/'
resized_folder = 'your-main-directory\CatVsDog Classifier/image resized/'

for i in range(2000):

  filename = os.listdir(original_folder)[i]
  img_path = original_folder+filename

  img = Image.open(img_path)
  img = img.resize((224, 224))
  img = img.convert('RGB')

  newImgPath = resized_folder+filename
  img.save(newImgPath)
```

In [11]:
```
# RESIZED DOG IMAGE
img = mpimg.imread('your-main-directory\CatVsDog Classifier/image resized/dog.342.jpg')
imgplt = plt.imshow(img)
plt.show()
```



In [12]:
```
# RESIZED CAT IMAGE
img = mpimg.imread('your-main-directory\CatVsDog Classifier/image resized/cat.658.jpg')
imgplt = plt.imshow(img)
plt.show()
```



Creating a Label for the Classification

0 = Cat
1 = Dog

ASSIGNING THE LABELS

In [13]:
```
filenames = os.listdir('your-main-directory\CatVsDog Classifier/image resized/')


labels = []

for i in range(2000):

  file_name = filenames[i]
  label = file_name[0:3]

  if label == 'dog':
    labels.append(1)

  else:
    labels.append(0)
```

In [14]:
```
print(filenames[0:5])
print(len(filenames))
```

```
['cat.0.jpg', 'cat.1.jpg', 'cat.10.jpg', 'cat.100.jpg', 'cat.101.jpg']
2000
```

In [15]:
```
print(labels[0:5])
print(len(labels))
```

```
[0, 0, 0, 0, 0]
2000
```

Counting the number of images of dogs and cats out of 2000 images

In [16]:
```
values, counts = np.unique(labels, return_counts=True)
print(values)
print(counts)
```

```
[0 1]
[1000 1000]
```

Converting Resized Images to Numpy arrays

In [17]:
```
import cv2
import glob
```

In [18]:
```
image_directory = 'your-main-directory\CatVsDog Classifier/image resized/'
image_extension = ['png', 'jpg']

files = []

[files.extend(glob.glob(image_directory + '*.' + e)) for e in image_extension]

dog_cat_images = np.asarray([cv2.imread(file) for file in files])
```

In [19]:
```
print(dog_cat_images)

type(dog_cat_images)
```

```
   ...
   [ 55 111  88]
   [ 54 110  87]
   [ 54 110  87]]]


 [[[155 143 139]
   [160 148 144]
   [157 145 141]
   ...
   [203 225 220]
   [207 230 225]
   [213 236 231]]

  [[136 124 120]
   [146 134 130]
   [153 140 138]
   ...
   [212 234 229]
```

In [20]:
```
print(dog_cat_images.shape)
```

```
(2000, 224, 224, 3)
```

Assigning the Labels to X & Y

In [21]:
```
X = dog_cat_images
Y = np.asarray(labels)
```

SPLITTING THE DATASET

Splitting the Dataset into Training Data and Testing Data

In [23]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2
```

In [26]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(2000, 224, 224, 3) (1600, 224, 224, 3) (400, 224, 224, 3)
```

training images = 1600

test images = 400

Scaling the Data

In [27]:
```
X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

In [28]:
```
print(X_train_scaled)
```

```
   ...
   [0.04313725 0.05098039 0.05490196]
   [0.03921569 0.04705882 0.05098039]
   [0.03137255 0.03921569 0.04313725]]

  [[0.21568627 0.21568627 0.21568627]
   [0.11764706 0.11764706 0.11764706]
   [0.08235294 0.07843137 0.08627451]
   ...
   [0.03921569 0.04705882 0.05098039]
   [0.03137255 0.03921569 0.04313725]
   [0.02352941 0.03137255 0.03529412]]

  [[0.23529412 0.23529412 0.23529412]
   [0.13333333 0.13333333 0.13333333]
   [0.09803922 0.09411765 0.10196078]
   ...
   [0.03529412 0.04313725 0.04705882]
   [0.02745098 0.03529412 0.03921569]
   [0.02352941 0.03137255 0.03529412]]]]
```

BUILDING THE NEURAL NETWORK

In [29]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in e:\anaconda\lib\site-packages (2.13.0)Not
e: you may need to restart the kernel to use updated packages.

Requirement already satisfied: tensorflow-intel==2.13.0 in e:\anaconda\lib\site-packag
es (from tensorflow) (2.13.0)
Requirement already satisfied: absl-py>=1.0.0 in e:\anaconda\lib\site-packages (from t
ensorflow-intel==2.13.0->tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in e:\anaconda\lib\site-packages (fro
m tensorflow-intel==2.13.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.1.21 in e:\anaconda\lib\site-packages
(from tensorflow-intel==2.13.0->tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in e:\anaconda\lib\site-packages (f
rom tensorflow-intel==2.13.0->tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in e:\anaconda\lib\site-packages (f
rom tensorflow-intel==2.13.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in e:\anaconda\lib\site-packages (from tens
orflow-intel==2.13.0->tensorflow) (3.7.0)
Requirement already satisfied: libclang>=13.0.0 in e:\anaconda\lib\site-packages (from
tensorflow-intel==2.13.0->tensorflow) (16.0.6)
Requirement already satisfied: numpy<=1.24.3,>=1.22 in e:\anaconda\lib\site-packages
(from tensorflow-intel==2.13.0->tensorflow) (1.24.3)
Requirement already satisfied: opt-einsum>=2.3.2 in e:\anaconda\lib\site-packages (fro
m tensorflow-intel==2.13.0->tensorflow) (3.3.0)
Requirement already satisfied: packaging in e:\anaconda\lib\site-packages (from tensor
flow-intel==2.13.0->tensorflow) (23.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=
4.21.5,<5.0.0dev,>=3.20.3 in e:\anaconda\lib\site-packages (from tensorflow-intel==2.1
3.0->tensorflow) (4.24.1)
Requirement already satisfied: setuptools in e:\anaconda\lib\site-packages (from tenso
rflow-intel==2.13.0->tensorflow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in e:\anaconda\lib\site-packages (from tens
orflow-intel==2.13.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in e:\anaconda\lib\site-packages (from
tensorflow-intel==2.13.0->tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in e:\anaconda\lib\site
-packages (from tensorflow-intel==2.13.0->tensorflow) (4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in e:\anaconda\lib\site-packages (from te
nsorflow-intel==2.13.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in e:\anaconda\lib\site-packages (f
rom tensorflow-intel==2.13.0->tensorflow) (1.57.0)
Requirement already satisfied: tensorboard<2.14,>=2.13 in e:\anaconda\lib\site-package
s (from tensorflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in e:\anaconda\lib\s
ite-packages (from tensorflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in e:\anaconda\lib\site-packages (f
rom tensorflow-intel==2.13.0->tensorflow) (2.13.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in e:\anaconda\lib
\site-packages (from tensorflow-intel==2.13.0->tensorflow) (0.31.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in e:\anaconda\lib\site-packages (fr
om astunparse>=1.6.0->tensorflow-intel==2.13.0->tensorflow) (0.38.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in e:\anaconda\lib\site-packages
(from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.22.0)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in e:\anaconda\lib\site-
packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in e:\anaconda\lib\site-packages (from
tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in e:\anaconda\lib\site-packages (f
rom tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in e:\anaconda\li
b\site-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(0.7.1)
```

```
Requirement already satisfied: werkzeug>=1.0.1 in e:\anaconda\lib\site-packages (from
tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.2.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in e:\anaconda\lib\site-packages
(from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensor
flow) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in e:\anaconda\lib\site-packages
(from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensor
flow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in e:\anaconda\lib\site-packages (from go
ogle-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(4.9)
Requirement already satisfied: urllib3<2.0 in e:\anaconda\lib\site-packages (from goog
le-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.2
6.16)
Requirement already satisfied: requests-oauthlib>=0.7.0 in e:\anaconda\lib\site-packag
es (from google-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow-intel==2.
13.0->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in e:\anaconda\lib\site-packag
es (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tenso
rflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in e:\anaconda\lib\site-packages (from req
uests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in e:\anaconda\lib\site-packages (fr
om requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in e:\anaconda\lib\site-packages (fro
m werkzeug>=1.0.1->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.
1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in e:\anaconda\lib\site-packages
(from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflo
w-intel==2.13.0->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in e:\anaconda\lib\site-packages (from
requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->ten
sorflow-intel==2.13.0->tensorflow) (3.2.2)
```

In [30]:
```
pip install tensorflow_hub
```

```
Requirement already satisfied: tensorflow_hub in e:\anaconda\lib\site-packages (0.14.
0)
Requirement already satisfied: numpy>=1.12.0 in e:\anaconda\lib\site-packages (from te
nsorflow_hub) (1.24.3)
Requirement already satisfied: protobuf>=3.19.6 in e:\anaconda\lib\site-packages (from
tensorflow_hub) (4.24.1)
Note: you may need to restart the kernel to use updated packages.
```

In [31]:
```
import tensorflow as tf
import tensorflow_hub as hub
```

We are using: **tf2-preview/mobilenet_v2/feature_vector**

> [TF2] Feature vectors of images with MobileNet V2 trained on ImageNet (ILSVRC-2012-CLS)

MobileNet V2 is a family of neural network architectures for efficient on-device image classification and related tasks, originally published by

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen: "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation", 2018. Mobilenets come in various sizes controlled by a multiplier for the depth (number of features) in the convolutional layers. They can also be trained for various sizes of input images to control inference speed.

This TF-Hub module uses the TF-Slim implementation of mobilenet_v2 with a depth multiplier of 1.0 and an input size of 224x224 pixels.

The module contains a trained instance of the network, packaged to get feature vectors from images. If you want the full model including the classification it was originally trained for, use module google/tf2-

In [32]:
```
mobilenet_model = 'https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4'

pretrained_model = hub.KerasLayer(mobilenet_model, input_shape=(224,224,3), trainable=F
```

In [33]:
```
num_of_classes = 2

model = tf.keras.Sequential([

    pretrained_model,
    tf.keras.layers.Dense(num_of_classes)

])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              2257984

 dense (Dense)               (None, 2)                 2562

=================================================================
Total params: 2260546 (8.62 MB)
Trainable params: 2562 (10.01 KB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

In [34]:
```
model.compile(
    optimizer = 'adam',
    loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics = ['acc']
)
```

Trainig the Model

In [35]:
```
model.fit(X_train_scaled, Y_train, epochs=5)
```

```
Epoch 1/5
50/50 [==============================] - 41s 747ms/step - loss: 0.2283 - acc: 0.9094
Epoch 2/5
50/50 [==============================] - 37s 730ms/step - loss: 0.0754 - acc: 0.9756
Epoch 3/5
50/50 [==============================] - 36s 725ms/step - loss: 0.0603 - acc: 0.9806
Epoch 4/5
50/50 [==============================] - 35s 708ms/step - loss: 0.0461 - acc: 0.9812
Epoch 5/5
50/50 [==============================] - 35s 696ms/step - loss: 0.0382 - acc: 0.9887
```

Out[35]:
```
<keras.src.callbacks.History at 0x2badc41dd50>
```

Accuracy Score

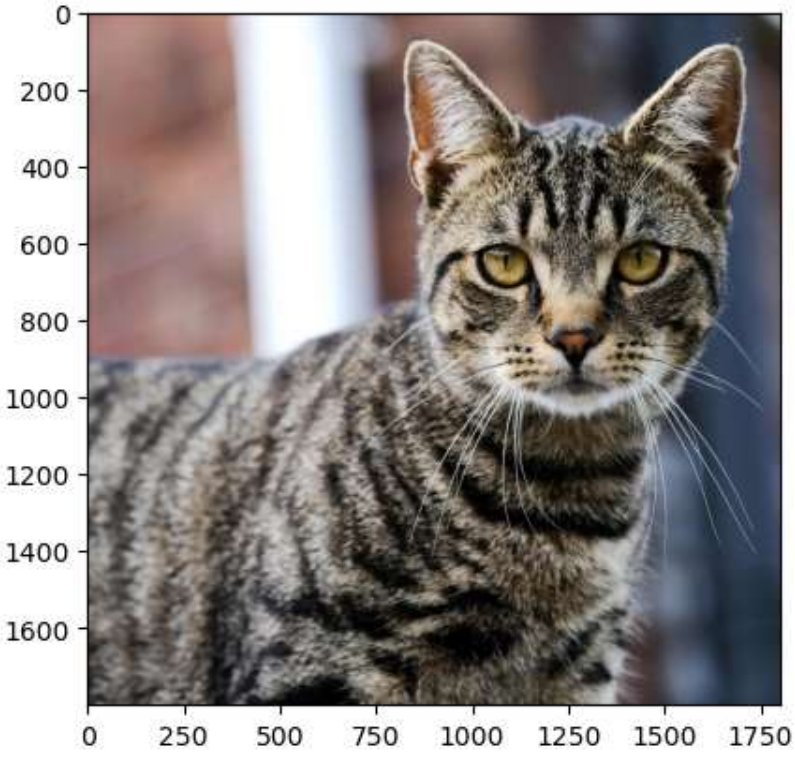
In [36]:
```
score, acc = model.evaluate(X_test_scaled, Y_test)
print('Test Loss =', score)
print('Test Accuracy =', acc)
print(acc*100, "%")
```

```
13/13 [==============================] - 10s 726ms/step - loss: 0.0488 - acc: 0.9825
Test Loss = 0.04876649007201195
Test Accuracy = 0.9825000166893005
98.25000166893005 %
```

THE PREDICTIVE MODEL

In [38]:
```
input_image_path = input('Path of the image to be predicted: ')

input_image = cv2.imread(input_image_path)

# cv2_imshow(input_image)
# cv2_imshow is specific to Google Collab environment

plt.imshow(cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB))

input_image_resize = cv2.resize(input_image, (224,224))

input_image_scaled = input_image_resize/255

image_reshaped = np.reshape(input_image_scaled, [1,224,224,3])

input_prediction = model.predict(image_reshaped)

print(input_prediction)

input_pred_label = np.argmax(input_prediction)

print(input_pred_label)

if input_pred_label == 0:
  print('The image if of a Cat')

else:
  print('The image is of a Dog')
```

```
Path of the image to be predicted: E:\Source Codes\AIML\CatVsDog Classifier\test data
\cat1.jpg
1/1 [==============================] - 0s 465ms/step
[[ 3.9810634 -3.4388103]]
0
The image if of a Cat
```


0
200
400
600
800
1000
1200
1400
1600
0
250
500
750
1000
1250
1500
1750

In [ ]: