Species_Classification_Charlie

June 30, 2023

Extracting dataset

[ ]:
```
!pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages
(1.5.13)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-
packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2022.12.7)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2.27.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from kaggle) (4.65.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-
packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-
packages (from kaggle) (1.26.15)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->kaggle) (3.4)
```

[ ]:
```
# configuring the path of Kaggle.json file
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Importing the Dog vs Cat Dataset from Kaggle

[ ]:
```
#Kaggle api
!kaggle competitions download -c dogs-vs-cats
```

```
Downloading dogs-vs-cats.zip to /content
100% 811M/812M [00:21<00:00, 42.1MB/s]
100% 812M/812M [00:21<00:00, 39.8MB/s]
```

[ ]:
```python
# extracting the compressed dataset
from zipfile import ZipFile

dataset = '/content/dogs-vs-cats.zip'

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

```
The dataset is extracted
```

[ ]:
```python
# extracting the compressed dataset
from zipfile import ZipFile

dataset = '/content/train.zip'

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

```
The dataset is extracted
```

[ ]:
```python
import os
# counting the number of files in train folder
path, dirs, files = next(os.walk('/content/train'))
file_count = len(files)
print('Number of images: ', file_count)
```

```
Number of images:  25000
```

Printing the name of images

[ ]:
```python
file_names = os.listdir('/content/train/')
print(file_names)
```

```
['dog.9890.jpg', 'cat.12164.jpg', 'dog.1148.jpg', 'dog.631.jpg', 'dog.521.jpg',
'cat.6056.jpg', 'cat.4778.jpg', 'dog.7353.jpg', 'dog.12044.jpg', 'cat.1649.jpg',
'cat.12026.jpg', 'cat.11705.jpg', 'dog.1235.jpg', 'dog.7931.jpg',
'cat.4980.jpg', 'dog.348.jpg', 'cat.4846.jpg', 'cat.5522.jpg', 'dog.6748.jpg',
'dog.5854.jpg', 'dog.5317.jpg', 'dog.757.jpg', 'dog.2496.jpg', 'dog.12154.jpg',
'cat.4279.jpg', 'cat.2253.jpg', 'cat.4070.jpg', 'dog.9629.jpg', 'dog.10618.jpg',
'cat.3524.jpg', 'dog.5657.jpg', 'dog.559.jpg', 'dog.1248.jpg', 'cat.6530.jpg',
'dog.10649.jpg', 'cat.9529.jpg', 'cat.6226.jpg', 'dog.10650.jpg', 'cat.702.jpg',
'dog.3110.jpg', 'cat.2257.jpg', 'dog.4863.jpg', 'dog.964.jpg', 'cat.10445.jpg',
'cat.6463.jpg', 'dog.5994.jpg', 'cat.4409.jpg', 'cat.7246.jpg', 'cat.6806.jpg',
'dog.11944.jpg', 'dog.6147.jpg', 'cat.4563.jpg', 'dog.6431.jpg', 'dog.3634.jpg',
```

```
'dog.10003.jpg', 'cat.3481.jpg', 'cat.21.jpg', 'cat.3845.jpg', 'dog.6588.jpg',
'cat.3320.jpg', 'cat.7607.jpg', 'cat.4097.jpg', 'cat.5032.jpg', 'dog.5652.jpg',
'dog.2490.jpg', 'dog.1930.jpg', 'dog.2661.jpg', 'cat.9689.jpg', 'dog.3801.jpg',
'cat.8145.jpg', 'cat.5823.jpg', 'cat.9495.jpg', 'cat.6241.jpg', 'dog.10399.jpg',
'dog.9688.jpg', 'cat.4682.jpg', 'dog.7392.jpg', 'cat.8997.jpg', 'cat.2626.jpg',
'dog.6044.jpg', 'dog.7853.jpg', 'dog.4100.jpg', 'cat.8473.jpg', 'cat.9909.jpg',
'dog.3872.jpg', 'dog.109.jpg', 'dog.987.jpg', 'dog.8283.jpg', 'cat.4931.jpg',
'dog.9774.jpg', 'cat.12019.jpg', 'dog.7715.jpg', 'cat.491.jpg', 'cat.2677.jpg',
'cat.9382.jpg', 'cat.2254.jpg', 'dog.8189.jpg', 'dog.2506.jpg', 'dog.3856.jpg',
'dog.2704.jpg', 'cat.883.jpg', 'cat.12445.jpg', 'dog.5119.jpg', 'dog.10185.jpg',
'cat.3864.jpg', 'dog.11402.jpg', 'dog.2451.jpg', 'dog.12090.jpg',
'dog.1065.jpg', 'dog.10416.jpg', 'cat.1854.jpg', 'cat.6156.jpg', 'cat.8196.jpg',
'dog.5988.jpg', 'dog.2387.jpg', 'dog.9867.jpg', 'cat.2005.jpg', 'cat.9798.jpg',
'dog.7043.jpg', 'dog.5734.jpg', 'cat.8924.jpg', 'cat.2902.jpg', 'cat.6807.jpg',
'cat.8330.jpg', 'cat.6031.jpg', 'cat.3827.jpg', 'dog.6029.jpg', 'cat.4854.jpg',
'dog.11423.jpg', 'dog.2702.jpg', 'dog.7096.jpg', 'cat.10074.jpg',
'cat.3775.jpg', 'cat.9509.jpg', 'dog.6109.jpg', 'dog.6661.jpg', 'cat.106.jpg',
'cat.4600.jpg', 'cat.10078.jpg', 'dog.8496.jpg', 'dog.5125.jpg',
'dog.10851.jpg', 'cat.7760.jpg', 'dog.11100.jpg', 'cat.10302.jpg',
'cat.7797.jpg', 'dog.1703.jpg', 'cat.8577.jpg', 'dog.12041.jpg', 'cat.67.jpg',
'dog.11638.jpg', 'dog.8547.jpg', 'dog.4610.jpg', 'dog.10444.jpg',
'cat.9596.jpg', 'cat.241.jpg', 'cat.7483.jpg', 'dog.5664.jpg', 'dog.7650.jpg',
'cat.8410.jpg', 'dog.8635.jpg', 'cat.9245.jpg', 'dog.11912.jpg', 'dog.8297.jpg',
'cat.2307.jpg', 'cat.11098.jpg', 'cat.5832.jpg', 'cat.709.jpg', 'cat.255.jpg',
'cat.8200.jpg', 'cat.9230.jpg', 'dog.1193.jpg', 'dog.10846.jpg', 'cat.6794.jpg',
'cat.699.jpg', 'cat.6255.jpg', 'dog.6738.jpg', 'cat.1235.jpg', 'cat.6616.jpg',
'dog.4119.jpg', 'cat.1983.jpg', 'dog.6929.jpg', 'cat.1113.jpg', 'dog.7484.jpg',
'cat.3766.jpg', 'cat.1352.jpg', 'dog.4937.jpg', 'cat.4701.jpg', 'cat.7719.jpg',
'dog.5493.jpg', 'dog.4153.jpg', 'dog.77.jpg', 'dog.1139.jpg', 'cat.3023.jpg',
'cat.10798.jpg', 'dog.4466.jpg', 'cat.10391.jpg', 'cat.3929.jpg',
'cat.4817.jpg', 'cat.9507.jpg', 'dog.9351.jpg', 'dog.10422.jpg', 'dog.7290.jpg',
'dog.10863.jpg', 'cat.5829.jpg', 'dog.7171.jpg', 'cat.11083.jpg',
'dog.5216.jpg', 'cat.5258.jpg', 'cat.845.jpg', 'dog.1046.jpg', 'cat.4315.jpg',
'cat.3062.jpg', 'dog.8023.jpg', 'cat.3896.jpg', 'cat.1895.jpg', 'dog.1578.jpg',
'cat.5466.jpg', 'cat.1334.jpg', 'dog.5613.jpg', 'dog.7848.jpg', 'cat.4343.jpg',
'cat.4946.jpg', 'dog.9384.jpg', 'cat.7015.jpg', 'dog.2962.jpg', 'dog.5870.jpg',
'dog.11864.jpg', 'cat.6468.jpg', 'cat.9347.jpg', 'cat.2440.jpg',
'cat.10342.jpg', 'cat.5952.jpg', 'dog.9112.jpg', 'cat.7564.jpg', 'dog.1869.jpg',
'cat.8064.jpg', 'cat.12120.jpg', 'cat.11045.jpg', 'dog.1643.jpg',
'dog.2664.jpg', 'dog.4800.jpg', 'dog.4388.jpg', 'dog.4147.jpg', 'cat.2130.jpg',
'dog.5031.jpg', 'dog.8049.jpg', 'cat.974.jpg', 'dog.2928.jpg', 'cat.11534.jpg',
'cat.4575.jpg', 'dog.10322.jpg']
```

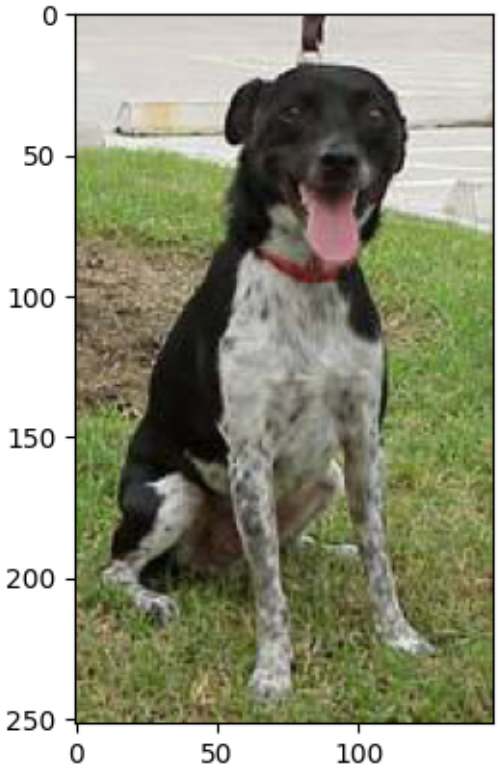
Importing the Dependencies

[ ]:
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

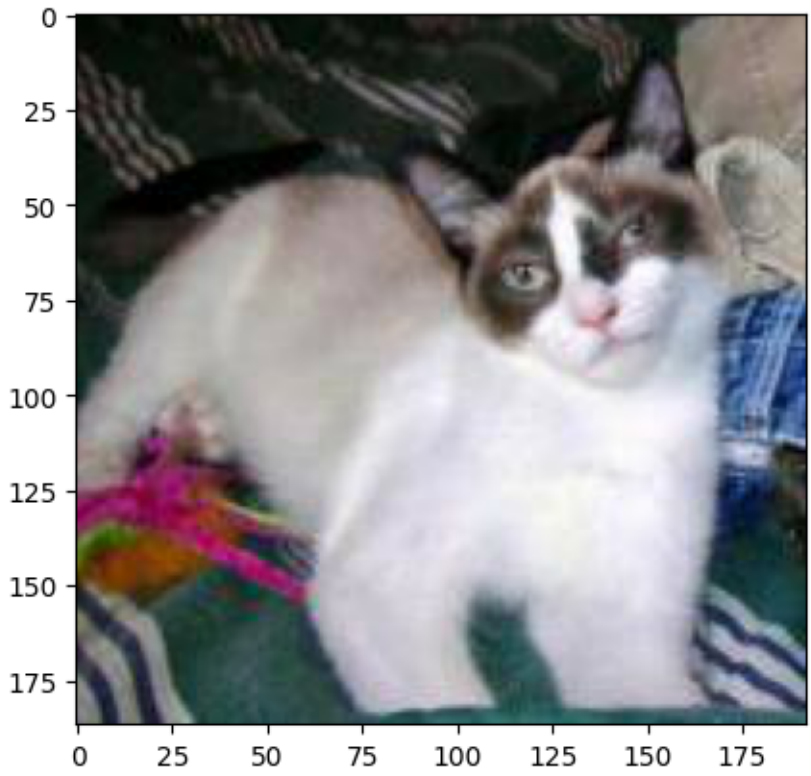
```
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
from google.colab.patches import cv2_imshow
```

Displaying the images of dogs and cats

[ ]:
```
# display dog image
img = mpimg.imread('/content/train/dog.9890.jpg')
imgplt = plt.imshow(img)
plt.show()
```



[ ]:
```
# display cat image
img = mpimg.imread('/content/train/cat.6056.jpg')
imgplt = plt.imshow(img)
plt.show()
```



[ ]:
```
file_names = os.listdir('/content/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)
```

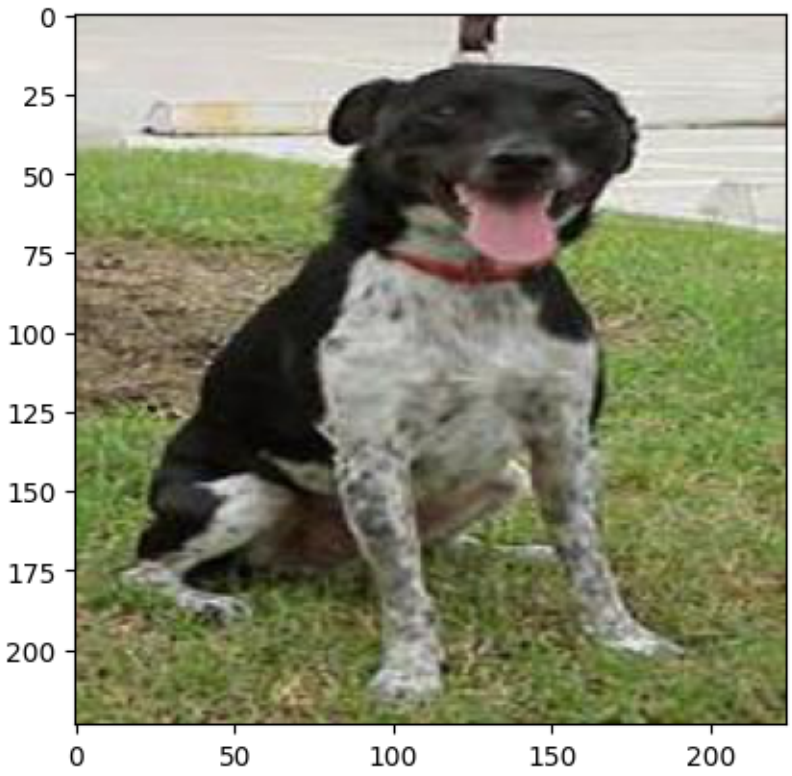
```
Number of dog images = 12500
Number of cat images = 12500
```

Resizing all the images

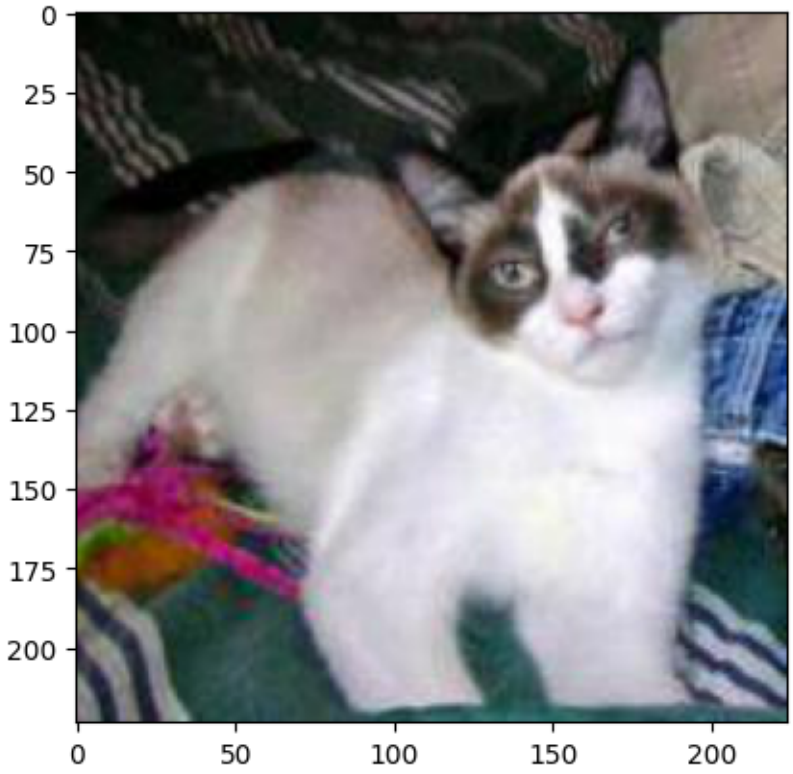
[ ]:
```
#creating a directory for resized images
os.mkdir('/content/image resized')
```

[ ]:
```
original_folder = '/content/train/'
resized_folder = '/content/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)
```

[ ]:
```
# display resized dog image
img = mpimg.imread('/content/image resized/dog.9890.jpg')
imgplt = plt.imshow(img)
plt.show()
```



[ ]:
```
# display resized cat image
img = mpimg.imread('/content/image resized/cat.6056.jpg')
imgplt = plt.imshow(img)
plt.show()
```



[ ]:
```
file_names = os.listdir('/content/image resized/')
print(file_names)
```

```
['dog.9890.jpg', 'cat.12164.jpg', 'dog.1148.jpg', 'dog.631.jpg', 'dog.521.jpg',
'cat.6056.jpg', 'cat.4778.jpg', 'dog.7353.jpg', 'dog.12044.jpg', 'cat.1649.jpg',
'cat.12026.jpg', 'cat.11705.jpg', 'dog.1235.jpg', 'dog.7931.jpg',
'cat.4980.jpg', 'dog.348.jpg', 'cat.4846.jpg', 'cat.5522.jpg', 'dog.6748.jpg',
'dog.5854.jpg', 'dog.5317.jpg', 'dog.757.jpg', 'dog.2496.jpg', 'dog.12154.jpg',
'cat.4279.jpg', 'cat.2253.jpg', 'cat.4070.jpg', 'dog.9629.jpg', 'dog.10618.jpg',
'cat.3524.jpg', 'dog.5657.jpg', 'dog.559.jpg', 'dog.1248.jpg', 'cat.6530.jpg',
'dog.10649.jpg', 'cat.9529.jpg', 'cat.6226.jpg', 'dog.10650.jpg', 'cat.702.jpg',
'dog.3110.jpg', 'cat.2257.jpg', 'dog.4863.jpg', 'dog.964.jpg', 'cat.10445.jpg',
'cat.6463.jpg', 'dog.5994.jpg', 'cat.4409.jpg', 'cat.7246.jpg', 'cat.6806.jpg',
'dog.11944.jpg', 'dog.6147.jpg', 'cat.4563.jpg', 'dog.6431.jpg', 'dog.3634.jpg',
'dog.1332.jpg', 'cat.2476.jpg', 'cat.4647.jpg', 'dog.5687.jpg', 'cat.5798.jpg',
'cat.10075.jpg', 'dog.11350.jpg', 'dog.10229.jpg', 'cat.4769.jpg',
'cat.4654.jpg', 'cat.7689.jpg', 'cat.11075.jpg', 'cat.5400.jpg', 'cat.6041.jpg',
```

```
'cat.9492.jpg', 'dog.11417.jpg', 'dog.3301.jpg', 'cat.12229.jpg',
'cat.1638.jpg', 'cat.7865.jpg', 'dog.11814.jpg', 'cat.1978.jpg', 'dog.2004.jpg',
'dog.2113.jpg', 'cat.7244.jpg', 'cat.12337.jpg', 'cat.5341.jpg',
'dog.11782.jpg', 'cat.11681.jpg', 'dog.47.jpg', 'dog.2756.jpg', 'dog.3963.jpg',
'dog.11171.jpg', 'cat.1878.jpg', 'cat.8455.jpg', 'cat.4926.jpg', 'cat.3947.jpg',
'dog.8118.jpg', 'cat.4938.jpg', 'dog.7481.jpg', 'cat.3080.jpg', 'dog.7849.jpg',
'cat.1879.jpg', 'dog.1727.jpg', 'dog.3063.jpg', 'dog.1457.jpg', 'cat.6258.jpg',
'dog.1515.jpg', 'cat.10613.jpg', 'cat.3883.jpg', 'cat.8299.jpg', 'cat.6815.jpg',
'dog.6178.jpg', 'cat.1175.jpg', 'dog.5048.jpg', 'cat.9469.jpg', 'dog.12107.jpg',
'cat.747.jpg', 'dog.9965.jpg', 'cat.11523.jpg', 'cat.8725.jpg', 'dog.7761.jpg',
'cat.4042.jpg', 'dog.10665.jpg', 'dog.270.jpg', 'cat.4174.jpg', 'dog.5157.jpg',
'dog.11903.jpg', 'dog.8620.jpg', 'dog.7169.jpg', 'dog.11253.jpg',
'dog.2569.jpg', 'cat.7318.jpg', 'cat.6104.jpg', 'cat.9157.jpg', 'cat.5120.jpg',
'dog.6252.jpg', 'cat.5721.jpg', 'dog.1818.jpg', 'cat.10126.jpg', 'cat.7381.jpg',
'dog.3471.jpg', 'cat.7993.jpg', 'cat.4289.jpg', 'cat.1462.jpg', 'cat.9754.jpg',
'cat.3047.jpg', 'dog.4469.jpg', 'dog.1499.jpg', 'dog.951.jpg', 'cat.3312.jpg',
'cat.7327.jpg', 'dog.5827.jpg', 'dog.6504.jpg', 'dog.4862.jpg', 'dog.11532.jpg',
'cat.4887.jpg', 'dog.7930.jpg', 'dog.6198.jpg', 'cat.5023.jpg', 'dog.654.jpg',
'cat.4590.jpg', 'dog.2528.jpg', 'cat.586.jpg', 'cat.4429.jpg', 'cat.3678.jpg',
'dog.10680.jpg', 'dog.1971.jpg', 'dog.10839.jpg', 'cat.8081.jpg',
'cat.3997.jpg', 'cat.5089.jpg', 'cat.8525.jpg', 'dog.1496.jpg', 'dog.7809.jpg',
'dog.12202.jpg']
```

Creating labels for resized images of dogs and cats

Cat –> 0

Dog –> 1

[ ]:
```
filenames = os.listdir('/content/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)
```

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

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

```
['dog.9890.jpg', 'cat.12164.jpg', 'dog.1148.jpg', 'dog.631.jpg', 'dog.521.jpg']
2000
[1, 0, 1, 1, 1]
```

2000

[ ]:
```
# counting the images of dogs and cats out of 2000 images
values, counts = np.unique(labels, return_counts=True)
print(values)
print(counts)
```

```
[0 1]
[ 975 1025]
```

Converting all the resized images to numpy arrays

[ ]:
```
import cv2
import glob
```

[ ]:
```
image_directory = '/content/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])
```

[ ]:
```
type(dog_cat_images)
```

[ ]:
```
numpy.ndarray
```

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

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

Spliting the data

[ ]:
```
X = dog_cat_images
y = np.asarray(labels)
```

Train Test Split

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=2)
```

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

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

So we have,

1600 –> training images

400 –> test images

[ ]:
```
# scaling the data
X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

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

```
[[[[0.0627451  0.07843137 0.17254902]
   [0.09019608 0.10588235 0.2       ]
   [0.08627451 0.10980392 0.2       ]
   …
   [0.03529412 0.04705882 0.06666667]
   [0.02352941 0.03529412 0.05490196]
   [0.03137255 0.04313725 0.0627451 ]]

  [[0.0745098  0.09019608 0.18823529]
   [0.09411765 0.10980392 0.20392157]
   [0.09019608 0.10980392 0.20784314]
   …
   [0.03529412 0.04705882 0.06666667]
   [0.02745098 0.03529412 0.06666667]
   [0.03529412 0.04705882 0.06666667]]

  [[0.08235294 0.09803922 0.20784314]
   [0.09803922 0.11764706 0.21568627]
   [0.09019608 0.10588235 0.21568627]
   …
   [0.04313725 0.05098039 0.08235294]
   [0.03529412 0.03921569 0.07843137]
   [0.03921569 0.04705882 0.07843137]]


  …

  [[0.08627451 0.0627451  0.06666667]
   [0.09019608 0.06666667 0.07058824]
   [0.09803922 0.0745098  0.07843137]
   …
   [0.01960784 0.02352941 0.00784314]
   [0.04705882 0.05098039 0.03529412]
   [0.01960784 0.02352941 0.00784314]]

  [[0.08627451 0.0627451  0.06666667]
   [0.09019608 0.06666667 0.07058824]
   [0.09803922 0.0745098  0.07843137]
   …
   [0.00392157 0.00784314 0.        ]
   [0.04705882 0.05098039 0.03529412]
   [0.04313725 0.04705882 0.03137255]]
```

```
 …

 [[0.16862745 0.17647059 0.21960784]
  [0.17647059 0.18431373 0.22352941]
  [0.17254902 0.18039216 0.21960784]
  …
  [0.62352941 0.59215686 0.59607843]
  [0.61568627 0.58431373 0.58823529]
  [0.60784314 0.57647059 0.58039216]]

 [[0.16862745 0.16862745 0.21568627]
  [0.18431373 0.18823529 0.22745098]
  [0.18039216 0.18823529 0.22745098]
  …
  [0.61176471 0.58039216 0.58431373]
  [0.60784314 0.57647059 0.58039216]
  [0.6        0.56862745 0.57254902]]

 [[0.08235294 0.08235294 0.12941176]
  [0.11372549 0.11764706 0.15686275]
  [0.12941176 0.1372549  0.17647059]
  …
  [0.60392157 0.57254902 0.57647059]
  [0.6        0.56862745 0.57254902]
  [0.59215686 0.56078431 0.56470588]]]
```

Building the Neural Network

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

[ ]:
```
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=False)
```

[ ]:
```
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: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

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

[ ]:
```
model.fit(X_train_scaled,y_train,epochs=5)
```

```
Epoch 1/5
50/50 [==============================] - 71s 1s/step - loss: 0.1895 - acc:
0.9237
Epoch 2/5
50/50 [==============================] - 66s 1s/step - loss: 0.0725 - acc:
0.9719
Epoch 3/5
50/50 [==============================] - 65s 1s/step - loss: 0.0553 - acc:
0.9800
Epoch 4/5
50/50 [==============================] - 83s 2s/step - loss: 0.0435 - acc:
0.9831
Epoch 5/5
50/50 [==============================] - 67s 1s/step - loss: 0.0379 - acc:
0.9887
```

[ ]:
```
<keras.callbacks.History at 0x7f507081d960>
```

[ ]:
```
score,acc = model.evaluate(X_test_scaled,y_test)
print('Test Loss = ',score)
print('Test Accuracy = ',acc)
```

```
13/13 [==============================] - 18s 1s/step - loss: 0.0673 - acc:
0.9725
Test Loss =  0.06727667897939682
Test Accuracy =  0.9725000262260437
```

Predictive System

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

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

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 represents a Cat')

else:
  print('The image represents a Dog')
```

```
Path of the image to be predicted: /content/dog.jpg
```

```
1/1 [==============================] - 1s 565ms/step
[[-4.3842177  5.3410845]]
1
The image represents a Dog
```

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

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

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)
```

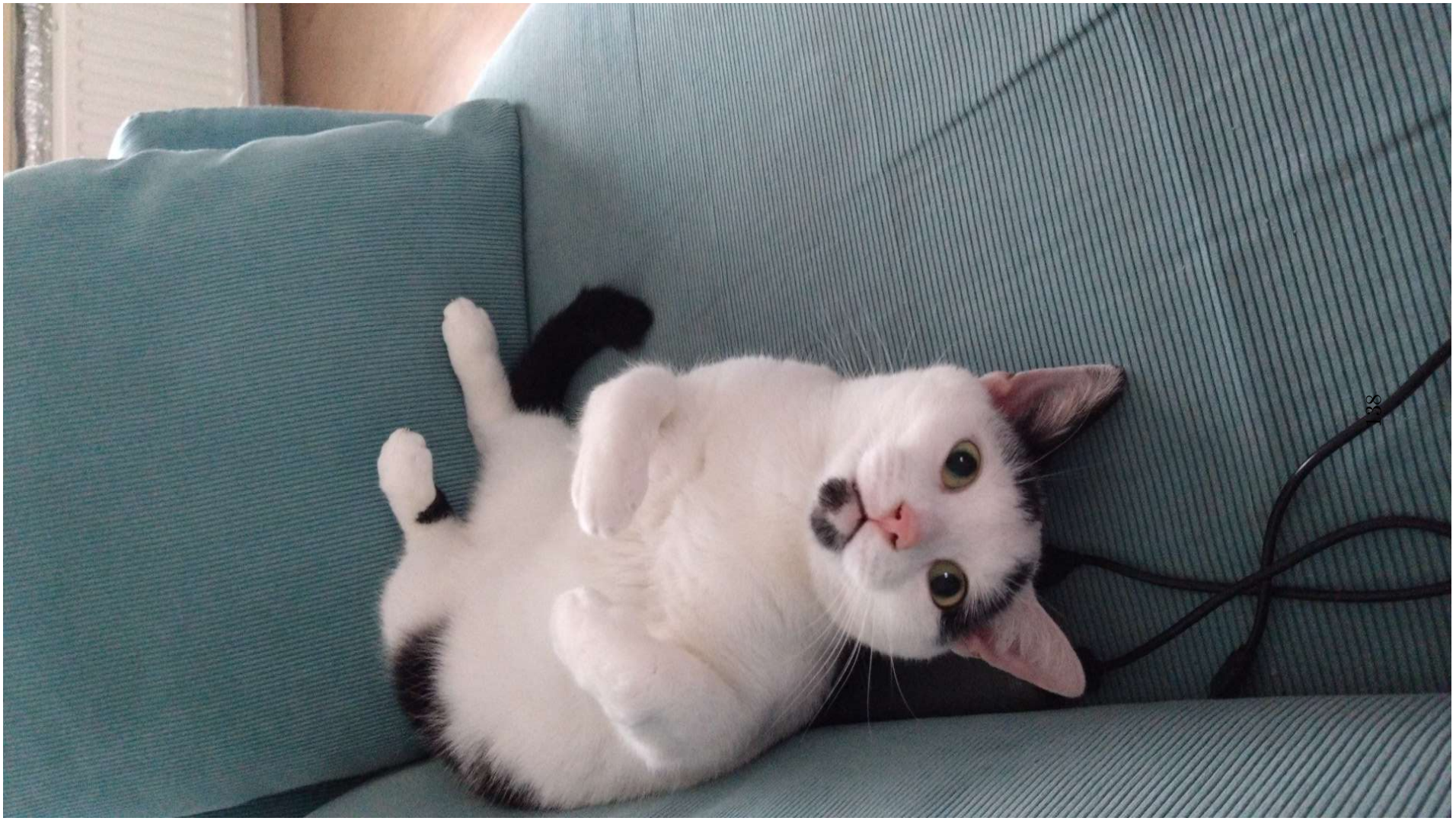
```
input_pred_label = np.argmax(input_prediction)

print(input_pred_label)

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

else:
  print('The image represents a Dog')
```

```
1/1 [==============================] - 0s 137ms/step
[[ 3.9385471 -3.4532409]]
0
The image represents a Cat
```

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

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

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 represents a Cat')

else:
  print('The image represents a Dog')
```

```
Path of the image to be predicted: /content/charlie.jpg
```



```
1/1 [==============================] - 0s 109ms/step
[[-0.41442183 -2.6167197 ]]
0
The image represents a Cat
```

[ ]: