In [ ]:
```
'''
    Author: A.Shrikant
'''
```

In [ ]:
```
# Basic libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
import os
import shutil  # shutil module provides a higher-level interface for file operations
```

In [ ]:
```
directory_path = '/content/dogs-vs-cats'

# Check if the directory exists before attempting to delete
if os.path.exists(directory_path):
  # Recursively delete a directory tree.
  shutil.rmtree(directory_path)
  print(f"Directory '{directory_path}' and its contents deleted.")
else:
    print(f"Directory '{directory_path}' does not exist.")
```

```
Directory '/content/dogs-vs-cats' does not exist.
```

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

In [ ]:
```
# !mkdir -p ~/.kaggle: This command creates a directory named .kaggle in the
# home directory (~). The -p flag ensures that if the directory already exists,
# it won't raise an error.

# !cp kaggle.json ~/.kaggle/: This command copies the Kaggle API key file
# (kaggle.json) to the .kaggle directory in the home directory (~). This is
# necessary because the Kaggle CLI (Command Line Interface) expects the API
# key to be present in the .kaggle directory.

# !chmod 600 ~/.kaggle/kaggle.json: This command sets the permissions of the
```

```
# kaggle.json file so that only the owner has read and write permissions. This
# is done for security reasons to restrict access to the Kaggle API key.
```

In [ ]:
```
!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
 98% 1.05G/1.06G [00:07<00:00, 302MB/s]
100% 1.06G/1.06G [00:07<00:00, 161MB/s]
```

In [ ]:
```
# import zipfile

# # Specify the path to the zip file
# zip_file_path = '/content/datasets/dataset-name.zip'

# # Specify the directory where you want to extract the contents
# extracted_folder_path = '/content/datasets/'

# # Open the zip file
# with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
#     # Extract all the contents to the specified directory
#     zip_ref.extractall(extracted_folder_path)

# # Now the contents of the zip file should be extracted to the specified directory
```

The command !unzip dataset-name.zip -d /content/datasets/ and the Python code using the zipfile module achieve the same goal of extracting the contents of a zip file to a specified directory. However, they differ in terms of execution and flexibility:

Execution Environment:

The !unzip command is a shell command that is executed directly in the command line of the Colab notebook. It's using the system's native unzip utility.

The Python code using the zipfile module is executed within a Python environment in a Colab cell.

Syntax and Flexibility:

The !unzip command is concise and specific to handling zip files. It's a command-line utility, and you provide the source zip file and the destination directory as arguments.

The Python code using the zipfile module is more flexible and can be integrated into a larger Python script or program. It provides programmatic control over the extraction process, allowing you to customize it further if needed.

In practical terms, for simple tasks like extracting a zip file in a Colab notebook, the !unzip command is often more straightforward and concise.

However, if you are working within a Python script or need more control over the extraction process, using the zipfile module provides additional flexibility and programmability. Choose the method that best suits your needs and the context of your workflow.

In [ ]:
```
# dogs-vs-cats.zip: This is the name of the ZIP file that you want
# to extract.

# 'dogs_vs_cats/*': This is an argument passed to unzip specifying which
# files or directories within the ZIP archive to extract. Here,
# dogs_vs_cats/* indicates that you want to extract all files and directories
# within the dogs_vs_cats directory that is inside the ZIP file.

# -d /content/: This option specifies the destination directory where the
# extracted files should be placed. In this case, /content/ is the root
# directory in Google Colab notebooks, often used for storing files and
# datasets.

# In a Google Colab notebook, each cell defaults to running Python code.
# However, if you prefix a line in a cell with an exclamation mark (!),
# it indicates that the line should be treated as a shell command
# (command-line interface command) rather than Python code. This allows you
# to execute command-line operations directly within the notebook environment.

!unzip dogs-vs-cats.zip 'dogs_vs_cats/*' -d /content/
```

```
Streaming output truncated to the last 5000 lines.
  inflating: /content/dogs_vs_cats/train/dogs/dog.4419.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.442.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4420.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4421.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4422.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4424.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4425.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4426.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4427.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4431.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4433.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4436.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4438.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4439.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.444.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4440.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4441.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4442.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4443.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4444.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4445.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4446.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.445.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4450.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4451.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4452.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4453.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4454.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4455.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4456.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4457.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4458.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4459.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.446.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4460.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4462.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4463.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4466.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4468.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.447.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.4470.jpg
```

```
  inflating: /content/dogs_vs_cats/train/dogs/dog.9997.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.9998.jpg
  inflating: /content/dogs_vs_cats/train/dogs/dog.9999.jpg
```

In [ ]:
```
!ls
```

```
dogs_vs_cats  dogs-vs-cats.zip  kaggle.json  sample_data
```

In [ ]:
```
!ls dogs_vs_cats
```

```
test  train
```

In [ ]:
```
#  In Unix-like operating systems, including Linux and macOS, the tilde is a
#  shorthand representation of the home directory.

# os.path.expanduser() takes a path string as an argument and returns a new
# string with the tilde expanded to the user's home directory.
# If the path does not start with a tilde, the function returns the input
# path unchanged.

kaggle_folder_path = os.path.expanduser("~/.kaggle")

if os.path.isdir(kaggle_folder_path):
    print("Directory found!")
else:
    print("Directory not found.")
```

```
Directory found!
```

In [ ]:
```
# Get the full path of the ~/.kaggle directory
full_path = os.path.expanduser("~/.kaggle")
print("Full path of ~/.kaggle:", full_path)
```

```
Full path of ~/.kaggle: /root/.kaggle
```

In [ ]:
```
# In Google Colab, the /root directory is not typically accessible. Colab runs
# in a managed environment, and users have access to specific directories for
# storing and managing files.

# However, this doesn't affect your ability to interact with the files and
# directories you have permission to use.

!ls /root
```

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import (Dense, Conv2D, MaxPooling2D, Flatten,
                          BatchNormalization, Dropout)
```

In [ ]:
```
# directory: Directory where the data is located.

# labels = "inferred" means it is expected that the in the directory where data
# is located there will be a sub-directory for each class in the target column.

# label_mode: String describing the encoding of labels. Options are:
#     - "int": means that the labels are encoded as integers
# (e.g. for sparse_categorical_crossentropy loss).

#     - "categorical" means that the labels are
# encoded as a categorical vector
# (e.g. for categorical_crossentropy loss).

#     - "binary" means that the labels (there can be only 2)
# are encoded as float32 scalars with values 0 or 1
# (e.g. for binary_crossentropy).

#     - None (no labels).

# batch_size: Size of the batches of data. This indicates how many images would
# be processed at a time during model training.

# image_size: Size to resize images to after they are read from disk,
# specified as (height, width).

train_dataset, validation_dataset = keras.utils.image_dataset_from_directory(
    directory='/content/dogs_vs_cats/train', labels = 'inferred',
    label_mode = 'int', batch_size = 32, image_size = (227,227), seed = 42,
    validation_split= 0.2, subset='both'
)
```

```
Found 20000 files belonging to 2 classes.
Using 16000 files for training.
Using 4000 files for validation.
```

In [ ]:
```
train_dataset
```

Out[ ]:
```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 227, 227, 3), dtype=tf.float32, name=None), TensorSpec(shape
=(None,), dtype=tf.int32, name=None))>
```

In [ ]:
```
# Finding number of images in each category in train_dataset:

def get_value_counts(tf_dataset):
  flag = True
  d = {}

  for images_batch in tf_dataset:
    _ , labels = images_batch

    a = np.unique(labels.numpy(), return_counts=True)

    for category in a[0]:
      if category in d:
        d[category] += a[1][category]
      else:
        d[category] = a[1][category]

  print(f'Number of images in each category: {d}')
```

In [ ]:
```
get_value_counts(train_dataset)
get_value_counts(validation_dataset)
```

```
Number of images in each category: {0: 8044, 1: 7956}
Number of images in each category: {0: 1956, 1: 2044}
```

In [ ]:
```
print(8044/7956)
print(1956/2044)
```

```
1.0110608345902463
0.9569471624266145
```

In [ ]:
```
type(train_dataset)
```

Out[ ]:

tensorflow.python.data.ops.prefetch_op._PrefetchDataset

def __init__(input_dataset, buffer_size, slack_period=None, name=None)

A `Dataset` that asynchronously prefetches its input.

In [ ]:
```
# Number of batches in train_dataset.
len(train_dataset)
```

Out[ ]:
```
500
```

In [ ]:
```
# Total images in train_dataset.
500*32
```

Out[ ]:
```
16000
```

In [ ]:
```
# Sample code to create train, validation and test dataset from a
# tensorflow-dataset when we do not have train and test directory containing
# images from different class in separate sub-directories.

# https://stackoverflow.com/questions/48213766/split-a-dataset-created-by-tensorflow-dataset-api-in-to-train-and-test
# tf.enable_eager_execution()

dataset = tf.data.Dataset.range(10)
print(type(dataset))

train_size = 2
valid_size = 3
test_size = 5

# reshuffle_each_iteration controls whether the shuffle order should be
# different for each epoch.
dataset = dataset.shuffle(buffer_size=10, reshuffle_each_iteration=False, seed=42)
train2 = dataset.take(train_size)
remaining2 = dataset.skip(train_size)
validation2= remaining2.take(valid_size)
test2 = remaining2.skip(valid_size)

for i in train2:
    print(i)
```

```
print('-'*30)

for i in validation2:
    print(i)

print('-'*30)

for i in test2:
    print(i)
```

```
<class 'tensorflow.python.data.ops.range_op._RangeDataset'>
tf.Tensor(2, shape=(), dtype=int64)
tf.Tensor(0, shape=(), dtype=int64)
------------------------------
tf.Tensor(7, shape=(), dtype=int64)
tf.Tensor(8, shape=(), dtype=int64)
tf.Tensor(9, shape=(), dtype=int64)
------------------------------
tf.Tensor(3, shape=(), dtype=int64)
tf.Tensor(6, shape=(), dtype=int64)
tf.Tensor(4, shape=(), dtype=int64)
tf.Tensor(1, shape=(), dtype=int64)
tf.Tensor(5, shape=(), dtype=int64)
```

In [ ]:
```
# Normalize the image data.
def process(image, label):
  # Casts a tensor to a new type.
  image = tf.cast(image/255, tf.float32)
  return image, label
```

In [ ]:
```
train_dataset = train_dataset.map(process)
validation_dataset = validation_dataset.map(process)
```

In [ ]:
```
test_dataset = keras.utils.image_dataset_from_directory(
    directory='/content/dogs_vs_cats/test', labels = 'inferred',
    label_mode = 'int', batch_size=32, image_size=(227,227)
)
test_dataset = test_dataset.map(process)
```

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

In [ ]:
```
get_value_counts(test_dataset)
```

```
Number of images in each category: {0: 2500, 1: 2500}
```

In [ ]:
```
num_elements_to_view = 1

# Create an iterator from the dataset
iterator = iter(train_dataset)

# Use the take method to get a specified number of elements
sample_elements = [next(iterator) for _ in range(num_elements_to_view)]
```

In [ ]:
```
sample_elements
```

Out[ ]:
```
[(<tf.Tensor: shape=(32, 227, 227, 3), dtype=float32, numpy=
  array([[[[0.54485214, 0.5487737 , 0.4860286 ],
           [0.3877626 , 0.39168417, 0.32893908],
           [0.37084848, 0.37477005, 0.31903884],
           ...,
           [0.44175044, 0.445672  , 0.4256495 ],
           [0.33142665, 0.33534822, 0.31072992],
           [0.30908778, 0.3156841 , 0.27628723]],

          [[0.5534787 , 0.5574003 , 0.49465516],
           [0.41925627, 0.42317784, 0.3642161 ],
           [0.37244   , 0.37636158, 0.3210304 ],
           ...,
           [0.38192406, 0.38584563, 0.36413142],
           [0.30985513, 0.31541005, 0.28020862],
           [0.357977  , 0.36582014, 0.3188995 ]],

          [[0.5215999 , 0.52552146, 0.4627763 ],
           [0.48587322, 0.4897948 , 0.43489283],
           [0.3860998 , 0.39002135, 0.3351194 ],
           ...,
           [0.32360578, 0.32752734, 0.30333117],
           [0.34674516, 0.35405272, 0.30728766],
           [0.38845584, 0.39629897, 0.34216577]],

          ...,

          [[0.18821082, 0.18821082, 0.15683827],
           [0.17676724, 0.17676724, 0.1453947 ],
           [0.19611338, 0.19611338, 0.16474083],
           ...,
           [0.16860226, 0.16468069, 0.14899442],
           [0.21345666, 0.20953509, 0.20062086],
           [0.21980806, 0.21588649, 0.20804335]],

          [[0.2044664 , 0.19837688, 0.16903417],
           [0.22167341, 0.2155839 , 0.1862412 ],
           [0.24633877, 0.24024926, 0.21090657],
           ...,
           [0.20938471, 0.20546314, 0.18977687],
           [0.29797333, 0.29405177, 0.28513753],
           [0.24110104, 0.23717947, 0.22933634]],
```

```
 [[0.20631155, 0.19454685, 0.16709587],
  [0.24205725, 0.23029254, 0.20284157],
  [0.25787827, 0.24611357, 0.21866259],
  ...,
  [0.26060283, 0.25668126, 0.24099496],
  [0.42075452, 0.41683295, 0.40791872],
  [0.32099792, 0.31707636, 0.30923322]]],


[[[0.6261553 , 0.5908612 , 0.46929255],
  [0.5623564 , 0.5270623 , 0.40549365],
  [0.5856353 , 0.5503412 , 0.42877257],
  ...,
  [0.7163428 , 0.7202644 , 0.6575193 ],
  [0.7176471 , 0.72156864, 0.65882355],
  [0.7176471 , 0.72156864, 0.65882355]],

 [[0.6261553 , 0.5908612 , 0.46929255],
  [0.5623564 , 0.5270623 , 0.40549365],
  [0.5856353 , 0.5503412 , 0.42877257],
  ...,
  [0.7163428 , 0.7202644 , 0.6575193 ],
  [0.7176471 , 0.72156864, 0.65882355],
  [0.7176471 , 0.72156864, 0.65882355]],

 [[0.6261553 , 0.5908612 , 0.46929255],
  [0.5623564 , 0.5270623 , 0.40549365],
  [0.5856353 , 0.5503412 , 0.42877257],
  ...,
  [0.7137255 , 0.7176471 , 0.654902  ],
  [0.7137255 , 0.7176471 , 0.654902  ],
  [0.7137255 , 0.7176471 , 0.654902  ]],

 ...,

 [[0.63116914, 0.48999268, 0.39381915],
  [0.582721  , 0.43527347, 0.34275374],
  [0.6325649 , 0.473082  , 0.37639308],
  ...,
  [0.5398315 , 0.48624268, 0.46140903],
  [0.51135015, 0.4682129 , 0.4462831 ],
```

```
  [0.4990545 , 0.45594248, 0.4402436 ]],

 [[0.631895  , 0.48700875, 0.422937  ],
  [0.56681263, 0.4231446 , 0.35037658],
  [0.5762563 , 0.4264816 , 0.33679175],
  ...,
  [0.52072173, 0.46607214, 0.44176888],
  [0.49017024, 0.44512686, 0.43007594],
  [0.47559363, 0.43704352, 0.4202294 ]],

 [[0.48490584, 0.33980778, 0.31314287],
  [0.53454375, 0.3925812 , 0.33533838],
  [0.57593566, 0.44129798, 0.33673722],
  ...,
  [0.50559145, 0.4441595 , 0.4219432 ],
  [0.48548847, 0.43372205, 0.41412282],
  [0.47372365, 0.42745098, 0.420653  ]]],


[[[0.578624  , 0.3903887 , 0.58646715],
  [0.6215754 , 0.43334007, 0.62941855],
  [0.49759093, 0.31645593, 0.5089842 ],
  ...,
  [0.26325837, 0.17772657, 0.12748888],
  [0.27628994, 0.20457865, 0.16480145],
  [0.3006913 , 0.23287258, 0.20448603]],

 [[0.61618704, 0.42795172, 0.6240302 ],
  [0.6113049 , 0.4230696 , 0.619148  ],
  [0.7070106 , 0.5258756 , 0.7184039 ],
  ...,
  [0.24945389, 0.15663417, 0.10872013],
  [0.24553862, 0.16493326, 0.12728953],
  [0.2408837 , 0.16906889, 0.1292399 ]],

 [[0.55893517, 0.37069988, 0.5667783 ],
  [0.70517015, 0.5169349 , 0.7130133 ],
  [0.5895018 , 0.40836677, 0.60089505],
  ...,
  [0.26999113, 0.17982206, 0.12494709],
  [0.26325724, 0.17023039, 0.12233356],
  [0.24769808, 0.16199368, 0.11158337]],
```

```
 ...,

 [[0.8096003 , 0.78999245, 0.76646304],
  [0.80336004, 0.7837522 , 0.7602228 ],
  [0.852248  , 0.8255399 , 0.8020105 ],
  ...,
  [0.26944718, 0.39579257, 0.4791193 ],
  [0.2515741 , 0.3776753 , 0.46831906],
  [0.3053066 , 0.43136695, 0.5294062 ]],

 [[0.8003372 , 0.78734607, 0.7605083 ],
  [0.78994405, 0.7760055 , 0.7496415 ],
  [0.8452558 , 0.8185476 , 0.7890282 ],
  ...,
  [0.2820229 , 0.41659603, 0.5124823 ],
  [0.2575815 , 0.39761934, 0.49775603],
  [0.32079074, 0.46278837, 0.5641359 ]],

 [[0.70227104, 0.7173589 , 0.68114907],
  [0.71922666, 0.7220174 , 0.68026304],
  [0.78168225, 0.75594527, 0.7164162 ],
  ...,
  [0.28962258, 0.4340369 , 0.53340405],
  [0.26090464, 0.42537794, 0.5296201 ],
  [0.30535218, 0.47766483, 0.5889433 ]]],


...,


[[[0.09135355, 0.04037315, 0.01684374],
  [0.09135355, 0.04037315, 0.01684374],
  [0.09135355, 0.04037315, 0.01684374],
  ...,
  [0.43309152, 0.37034643, 0.27230722],
  [0.43475857, 0.37201348, 0.28181738],
  [0.43475857, 0.37201348, 0.28181738]],

 [[0.11488296, 0.06390256, 0.04037315],
  [0.11488296, 0.06390256, 0.04037315],
  [0.11488296, 0.06390256, 0.04037315],
```

```
  ...,
  [0.44429472, 0.3815496 , 0.2835104 ],
  [0.4419798 , 0.3792347 , 0.28903863],
  [0.4419798 , 0.3792347 , 0.28903863]],

 [[0.10856008, 0.05757969, 0.02620713],
  [0.10856008, 0.05757969, 0.02620713],
  [0.10856008, 0.05757969, 0.02620713],
  ...,
  [0.4503626 , 0.3876175 , 0.2895783 ],
  [0.43467218, 0.37192708, 0.281731  ],
  [0.43467218, 0.37192708, 0.281731  ]],

 ...,

 [[0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.5019135 , 0.37517938, 0.27594817],
  [0.5544967 , 0.45953274, 0.37556463],
  [0.6041851 , 0.52183217, 0.44732237]],

 [[0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.44124848, 0.32916316, 0.24065153],
  [0.47073197, 0.38522625, 0.30441093],
  [0.49124923, 0.4088963 , 0.3343865 ]],

 [[0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.5053434 , 0.41174996, 0.33789068],
  [0.5621985 , 0.47848442, 0.40125233],
  [0.5303601 , 0.4480071 , 0.3734973 ]]],


[[[0.13912977, 0.10562367, 0.0667535 ],
  [0.19454591, 0.14356552, 0.11562354],
```

```
 [0.28072995, 0.21006395, 0.19445542],
 ...,
 [0.41174102, 0.38429004, 0.4156626 ],
 [0.4103222 , 0.3828712 , 0.41424376],
 [0.4103222 , 0.3828712 , 0.41424376]],

[[0.12472143, 0.08942731, 0.06693444],
 [0.19960491, 0.14498574, 0.13237268],
 [0.28388956, 0.21319768, 0.21038355],
 ...,
 [0.42709693, 0.39964595, 0.4310185 ],
 [0.43633068, 0.4088797 , 0.44025224],
 [0.43633068, 0.4088797 , 0.44025224]],

[[0.13572055, 0.09537877, 0.10048462],
 [0.18656045, 0.13110477, 0.13188396],
 [0.28537756, 0.21187977, 0.21756208],
 ...,
 [0.44881964, 0.42136866, 0.4527412 ],
 [0.4459618 , 0.41851082, 0.44988337],
 [0.4459618 , 0.41851082, 0.44988337]],

...,

[[0.77188945, 0.7091443 , 0.71306586],
 [0.77057016, 0.70782506, 0.71174663],
 [0.7653179 , 0.7025728 , 0.7064944 ],
 ...,
 [0.17687763, 0.13766195, 0.1807992 ],
 [0.10840502, 0.08249165, 0.11232659],
 [0.06310218, 0.04349433, 0.06702375]],

[[0.7580204 , 0.6952753 , 0.6991969 ],
 [0.74622977, 0.6834847 , 0.68740624],
 [0.74075425, 0.67800915, 0.6819307 ],
 ...,
 [0.18393657, 0.14472088, 0.18785813],
 [0.14661805, 0.11916707, 0.15053962],
 [0.09570695, 0.06825598, 0.09962852]],

[[0.7560681 , 0.693323  , 0.6972446 ],
 [0.7458668 , 0.6831217 , 0.68704325],
```

```
  [0.74502033, 0.68227524, 0.6861968 ],
  ...,
  [0.14927931, 0.11006362, 0.15320088],
  [0.14125402, 0.10203833, 0.14517559],
  [0.09570695, 0.05649127, 0.09962852]]],


[[[0.7417522 , 0.76136005, 0.73783064],
  [0.7422303 , 0.76183814, 0.7383087 ],
  [0.73213565, 0.7517435 , 0.7282141 ],
  ...,
  [0.6580224 , 0.7168459 , 0.7442969 ],
  [0.6684304 , 0.71516097, 0.7557586 ],
  [0.27425608, 0.29642072, 0.3681203 ]],

 [[0.715781  , 0.7353888 , 0.7118594 ],
  [0.6900411 , 0.70964897, 0.68611956],
  [0.71768236, 0.7372902 , 0.7137608 ],
  ...,
  [0.70507234, 0.75622547, 0.7875116 ],
  [0.6110452 , 0.66117847, 0.69890904],
  [0.29993805, 0.346565  , 0.39711347]],

 [[0.65902   , 0.67862785, 0.65509844],
  [0.6552463 , 0.67485416, 0.65132475],
  [0.67574084, 0.6953487 , 0.67181927],
  ...,
  [0.7525278 , 0.7985673 , 0.8171903 ],
  [0.5073535 , 0.56063026, 0.5878567 ],
  [0.3488911 , 0.4204986 , 0.4490841 ]],

 ...,

 [[0.01176471, 0.01176471, 0.01176471],
  [0.01176471, 0.01176471, 0.01176471],
  [0.01176471, 0.01176471, 0.01176471],
  ...,
  [0.6804633 , 0.6686986 , 0.6334045 ],
  [0.6778434 , 0.66846263, 0.63237387],
  [0.68695366, 0.68695366, 0.647738  ]],

 [[0.01176471, 0.01176471, 0.01176471],
```

```
          [0.01176471, 0.01176471, 0.01176471],
          [0.01176471, 0.01176471, 0.01176471],
          ...,
          [0.66664946, 0.65488476, 0.6195906 ],
          [0.6588749 , 0.6494942 , 0.6134054 ],
          [0.66423076, 0.66423076, 0.6250151 ]],

         [[0.01176471, 0.01176471, 0.01176471],
          [0.01176471, 0.01176471, 0.01176471],
          [0.01176471, 0.01176471, 0.01176471],
          ...,
          [0.66664946, 0.65488476, 0.6195906 ],
          [0.6611558 , 0.65177506, 0.6156863 ],
          [0.6603956 , 0.6603956 , 0.62117994]]]], dtype=float32)>,
 <tf.Tensor: shape=(32,), dtype=int32, numpy=
 array([1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 0, 1, 1, 0, 0, 1, 0, 1], dtype=int32)>)]
```

In [ ]:
```
len(sample_elements)
```

Out[ ]:
```
1
```

In [ ]:
```
type(sample_elements)
```

Out[ ]:
```
list
```

In [ ]:
```
type(sample_elements[0])
```

Out[ ]:
```
tuple
```

In [ ]:
```
sample_elements[0]
```

Out[ ]:
```
(<tf.Tensor: shape=(32, 227, 227, 3), dtype=float32, numpy=
 array([[[[0.54485214, 0.5487737 , 0.4860286 ],
          [0.3877626 , 0.39168417, 0.32893908],
          [0.37084848, 0.37477005, 0.31903884],
          ...,
          [0.44175044, 0.445672  , 0.4256495 ],
          [0.33142665, 0.33534822, 0.31072992],
          [0.30908778, 0.3156841 , 0.27628723]],

         [[0.5534787 , 0.5574003 , 0.49465516],
          [0.41925627, 0.42317784, 0.3642161 ],
          [0.37244   , 0.37636158, 0.3210304 ],
          ...,
          [0.38192406, 0.38584563, 0.36413142],
          [0.30985513, 0.31541005, 0.28020862],
          [0.357977  , 0.36582014, 0.3188995 ]],

         [[0.5215999 , 0.52552146, 0.4627763 ],
          [0.48587322, 0.4897948 , 0.43489283],
          [0.3860998 , 0.39002135, 0.3351194 ],
          ...,
          [0.32360578, 0.32752734, 0.30333117],
          [0.34674516, 0.35405272, 0.30728766],
          [0.38845584, 0.39629897, 0.34216577]],

         ...,

         [[0.18821082, 0.18821082, 0.15683827],
          [0.17676724, 0.17676724, 0.1453947 ],
          [0.19611338, 0.19611338, 0.16474083],
          ...,
          [0.16860226, 0.16468069, 0.14899442],
          [0.21345666, 0.20953509, 0.20062086],
          [0.21980806, 0.21588649, 0.20804335]],

         [[0.2044664 , 0.19837688, 0.16903417],
          [0.22167341, 0.2155839 , 0.1862412 ],
          [0.24633877, 0.24024926, 0.21090657],
          ...,
          [0.20938471, 0.20546314, 0.18977687],
          [0.29797333, 0.29405177, 0.28513753],
          [0.24110104, 0.23717947, 0.22933634]],
```

```
 [[0.20631155, 0.19454685, 0.16709587],
  [0.24205725, 0.23029254, 0.20284157],
  [0.25787827, 0.24611357, 0.21866259],
  ...,
  [0.26060283, 0.25668126, 0.24099496],
  [0.42075452, 0.41683295, 0.40791872],
  [0.32099792, 0.31707636, 0.30923322]]],


[[[0.6261553 , 0.5908612 , 0.46929255],
  [0.5623564 , 0.5270623 , 0.40549365],
  [0.5856353 , 0.5503412 , 0.42877257],
  ...,
  [0.7163428 , 0.7202644 , 0.6575193 ],
  [0.7176471 , 0.72156864, 0.65882355],
  [0.7176471 , 0.72156864, 0.65882355]],

 [[0.6261553 , 0.5908612 , 0.46929255],
  [0.5623564 , 0.5270623 , 0.40549365],
  [0.5856353 , 0.5503412 , 0.42877257],
  ...,
  [0.7163428 , 0.7202644 , 0.6575193 ],
  [0.7176471 , 0.72156864, 0.65882355],
  [0.7176471 , 0.72156864, 0.65882355]],

 [[0.6261553 , 0.5908612 , 0.46929255],
  [0.5623564 , 0.5270623 , 0.40549365],
  [0.5856353 , 0.5503412 , 0.42877257],
  ...,
  [0.7137255 , 0.7176471 , 0.654902  ],
  [0.7137255 , 0.7176471 , 0.654902  ],
  [0.7137255 , 0.7176471 , 0.654902  ]],

 ...,

 [[0.63116914, 0.48999268, 0.39381915],
  [0.582721  , 0.43527347, 0.34275374],
  [0.6325649 , 0.473082  , 0.37639308],
  ...,
  [0.5398315 , 0.48624268, 0.46140903],
  [0.51135015, 0.4682129 , 0.4462831 ],
```

```
  [0.4990545 , 0.45594248, 0.4402436 ]],

 [[0.631895  , 0.48700875, 0.422937  ],
  [0.56681263, 0.4231446 , 0.35037658],
  [0.5762563 , 0.4264816 , 0.33679175],
  ...,
  [0.52072173, 0.46607214, 0.44176888],
  [0.49017024, 0.44512686, 0.43007594],
  [0.47559363, 0.43704352, 0.4202294 ]],

 [[0.48490584, 0.33980778, 0.31314287],
  [0.53454375, 0.3925812 , 0.33533838],
  [0.57593566, 0.44129798, 0.33673722],
  ...,
  [0.50559145, 0.4441595 , 0.4219432 ],
  [0.48548847, 0.43372205, 0.41412282],
  [0.47372365, 0.42745098, 0.420653  ]]],


[[[0.578624  , 0.3903887 , 0.58646715],
  [0.6215754 , 0.43334007, 0.62941855],
  [0.49759093, 0.31645593, 0.5089842 ],
  ...,
  [0.26325837, 0.17772657, 0.12748888],
  [0.27628994, 0.20457865, 0.16480145],
  [0.3006913 , 0.23287258, 0.20448603]],

 [[0.61618704, 0.42795172, 0.6240302 ],
  [0.6113049 , 0.4230696 , 0.619148  ],
  [0.7070106 , 0.5258756 , 0.7184039 ],
  ...,
  [0.24945389, 0.15663417, 0.10872013],
  [0.24553862, 0.16493326, 0.12728953],
  [0.2408837 , 0.16906889, 0.1292399 ]],

 [[0.55893517, 0.37069988, 0.5667783 ],
  [0.70517015, 0.5169349 , 0.7130133 ],
  [0.5895018 , 0.40836677, 0.60089505],
  ...,
  [0.26999113, 0.17982206, 0.12494709],
  [0.26325724, 0.17023039, 0.12233356],
  [0.24769808, 0.16199368, 0.11158337]],
```

```
 ...,

 [[0.8096003 , 0.78999245, 0.76646304],
  [0.80336004, 0.7837522 , 0.7602228 ],
  [0.852248  , 0.8255399 , 0.8020105 ],
  ...,
  [0.26944718, 0.39579257, 0.4791193 ],
  [0.2515741 , 0.3776753 , 0.46831906],
  [0.3053066 , 0.43136695, 0.5294062 ]],

 [[0.8003372 , 0.78734607, 0.7605083 ],
  [0.78994405, 0.7760055 , 0.7496415 ],
  [0.8452558 , 0.8185476 , 0.7890282 ],
  ...,
  [0.2820229 , 0.41659603, 0.5124823 ],
  [0.2575815 , 0.39761934, 0.49775603],
  [0.32079074, 0.46278837, 0.5641359 ]],

 [[0.70227104, 0.7173589 , 0.68114907],
  [0.71922666, 0.7220174 , 0.68026304],
  [0.78168225, 0.75594527, 0.7164162 ],
  ...,
  [0.28962258, 0.4340369 , 0.53340405],
  [0.26090464, 0.42537794, 0.5296201 ],
  [0.30535218, 0.47766483, 0.5889433 ]]],


...,


[[[0.09135355, 0.04037315, 0.01684374],
  [0.09135355, 0.04037315, 0.01684374],
  [0.09135355, 0.04037315, 0.01684374],
  ...,
  [0.43309152, 0.37034643, 0.27230722],
  [0.43475857, 0.37201348, 0.28181738],
  [0.43475857, 0.37201348, 0.28181738]],

 [[0.11488296, 0.06390256, 0.04037315],
  [0.11488296, 0.06390256, 0.04037315],
  [0.11488296, 0.06390256, 0.04037315],
```

```
  ...,
  [0.44429472, 0.3815496 , 0.2835104 ],
  [0.4419798 , 0.3792347 , 0.28903863],
  [0.4419798 , 0.3792347 , 0.28903863]],

 [[0.10856008, 0.05757969, 0.02620713],
  [0.10856008, 0.05757969, 0.02620713],
  [0.10856008, 0.05757969, 0.02620713],
  ...,
  [0.4503626 , 0.3876175 , 0.2895783 ],
  [0.43467218, 0.37192708, 0.281731  ],
  [0.43467218, 0.37192708, 0.281731  ]],

 ...,

 [[0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.5019135 , 0.37517938, 0.27594817],
  [0.5544967 , 0.45953274, 0.37556463],
  [0.6041851 , 0.52183217, 0.44732237]],

 [[0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.44124848, 0.32916316, 0.24065153],
  [0.47073197, 0.38522625, 0.30441093],
  [0.49124923, 0.4088963 , 0.3343865 ]],

 [[0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.5053434 , 0.41174996, 0.33789068],
  [0.5621985 , 0.47848442, 0.40125233],
  [0.5303601 , 0.4480071 , 0.3734973 ]]],


[[[0.13912977, 0.10562367, 0.0667535 ],
  [0.19454591, 0.14356552, 0.11562354],
```

```
 [0.28072995, 0.21006395, 0.19445542],
 ...,
 [0.41174102, 0.38429004, 0.4156626 ],
 [0.4103222 , 0.3828712 , 0.41424376],
 [0.4103222 , 0.3828712 , 0.41424376]],

[[0.12472143, 0.08942731, 0.06693444],
 [0.19960491, 0.14498574, 0.13237268],
 [0.28388956, 0.21319768, 0.21038355],
 ...,
 [0.42709693, 0.39964595, 0.4310185 ],
 [0.43633068, 0.4088797 , 0.44025224],
 [0.43633068, 0.4088797 , 0.44025224]],

[[0.13572055, 0.09537877, 0.10048462],
 [0.18656045, 0.13110477, 0.13188396],
 [0.28537756, 0.21187977, 0.21756208],
 ...,
 [0.44881964, 0.42136866, 0.4527412 ],
 [0.4459618 , 0.41851082, 0.44988337],
 [0.4459618 , 0.41851082, 0.44988337]],

...,

[[0.77188945, 0.7091443 , 0.71306586],
 [0.77057016, 0.70782506, 0.71174663],
 [0.7653179 , 0.7025728 , 0.7064944 ],
 ...,
 [0.17687763, 0.13766195, 0.1807992 ],
 [0.10840502, 0.08249165, 0.11232659],
 [0.06310218, 0.04349433, 0.06702375]],

[[0.7580204 , 0.6952753 , 0.6991969 ],
 [0.74622977, 0.6834847 , 0.68740624],
 [0.74075425, 0.67800915, 0.6819307 ],
 ...,
 [0.18393657, 0.14472088, 0.18785813],
 [0.14661805, 0.11916707, 0.15053962],
 [0.09570695, 0.06825598, 0.09962852]],

[[0.7560681 , 0.693323  , 0.6972446 ],
 [0.7458668 , 0.6831217 , 0.68704325],
```

```
  [0.74502033, 0.68227524, 0.6861968 ],
  ...,
  [0.14927931, 0.11006362, 0.15320088],
  [0.14125402, 0.10203833, 0.14517559],
  [0.09570695, 0.05649127, 0.09962852]]],


[[[0.7417522 , 0.76136005, 0.73783064],
  [0.7422303 , 0.76183814, 0.7383087 ],
  [0.73213565, 0.7517435 , 0.7282141 ],
  ...,
  [0.6580224 , 0.7168459 , 0.7442969 ],
  [0.6684304 , 0.71516097, 0.7557586 ],
  [0.27425608, 0.29642072, 0.3681203 ]],

 [[0.715781  , 0.7353888 , 0.7118594 ],
  [0.6900411 , 0.70964897, 0.68611956],
  [0.71768236, 0.7372902 , 0.7137608 ],
  ...,
  [0.70507234, 0.75622547, 0.7875116 ],
  [0.6110452 , 0.66117847, 0.69890904],
  [0.29993805, 0.346565  , 0.39711347]],

 [[0.65902   , 0.67862785, 0.65509844],
  [0.6552463 , 0.67485416, 0.65132475],
  [0.67574084, 0.6953487 , 0.67181927],
  ...,
  [0.7525278 , 0.7985673 , 0.8171903 ],
  [0.5073535 , 0.56063026, 0.5878567 ],
  [0.3488911 , 0.4204986 , 0.4490841 ]],

 ...,

 [[0.01176471, 0.01176471, 0.01176471],
  [0.01176471, 0.01176471, 0.01176471],
  [0.01176471, 0.01176471, 0.01176471],
  ...,
  [0.6804633 , 0.6686986 , 0.6334045 ],
  [0.6778434 , 0.66846263, 0.63237387],
  [0.68695366, 0.68695366, 0.647738  ]],

 [[0.01176471, 0.01176471, 0.01176471],
```

```
          [0.01176471, 0.01176471, 0.01176471],
          [0.01176471, 0.01176471, 0.01176471],
          ...,
          [0.66664946, 0.65488476, 0.6195906 ],
          [0.6588749 , 0.6494942 , 0.6134054 ],
          [0.66423076, 0.66423076, 0.6250151 ]],

         [[0.01176471, 0.01176471, 0.01176471],
          [0.01176471, 0.01176471, 0.01176471],
          [0.01176471, 0.01176471, 0.01176471],
          ...,
          [0.66664946, 0.65488476, 0.6195906 ],
          [0.6611558 , 0.65177506, 0.6156863 ],
          [0.6603956 , 0.6603956 , 0.62117994]]]], dtype=float32)>,
 <tf.Tensor: shape=(32,), dtype=int32, numpy=
 array([1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 0, 1, 1, 0, 0, 1, 0, 1], dtype=int32)>)
```

In [ ]:
```
type(sample_elements[0][0])
```

Out[ ]:
```
tensorflow.python.framework.ops.EagerTensor
```

In [ ]:
```
obj = sample_elements[0][0]
print(obj.shape)
# obj
```

```
(32, 227, 227, 3)
```

In [ ]:
```
a = obj.numpy()
print(a.shape)
print(a)
```

```
(32, 227, 227, 3)
[[[[0.54485214 0.5487737  0.4860286 ]
   [0.3877626  0.39168417 0.32893908]
   [0.37084848 0.37477005 0.31903884]
   ...
   [0.44175044 0.445672   0.4256495 ]
   [0.33142665 0.33534822 0.31072992]
   [0.30908778 0.3156841  0.27628723]]

  [[0.5534787  0.5574003  0.49465516]
   [0.41925627 0.42317784 0.3642161 ]
   [0.37244    0.37636158 0.3210304 ]
   ...
   [0.38192406 0.38584563 0.36413142]
   [0.30985513 0.31541005 0.28020862]
   [0.357977   0.36582014 0.3188995 ]]

  [[0.5215999  0.52552146 0.4627763 ]
   [0.48587322 0.4897948  0.43489283]
   [0.3860998  0.39002135 0.3351194 ]
   ...
   [0.32360578 0.32752734 0.30333117]
   [0.34674516 0.35405272 0.30728766]
   [0.38845584 0.39629897 0.34216577]]

  ...

  [[0.18821082 0.18821082 0.15683827]
   [0.17676724 0.17676724 0.1453947 ]
   [0.19611338 0.19611338 0.16474083]
   ...
   [0.16860226 0.16468069 0.14899442]
   [0.21345666 0.20953509 0.20062086]
   [0.21980806 0.21588649 0.20804335]]

  [[0.2044664  0.19837688 0.16903417]
   [0.22167341 0.2155839  0.1862412 ]
   [0.24633877 0.24024926 0.21090657]
   ...
   [0.20938471 0.20546314 0.18977687]
   [0.29797333 0.29405177 0.28513753]
   [0.24110104 0.23717947 0.22933634]]
```

```
 [[0.20631155 0.19454685 0.16709587]
  [0.24205725 0.23029254 0.20284157]
  [0.25787827 0.24611357 0.21866259]
  ...
  [0.26060283 0.25668126 0.24099496]
  [0.42075452 0.41683295 0.40791872]
  [0.32099792 0.31707636 0.30923322]]]


[[[0.6261553  0.5908612  0.46929255]
  [0.5623564  0.5270623  0.40549365]
  [0.5856353  0.5503412  0.42877257]
  ...
  [0.7163428  0.7202644  0.6575193 ]
  [0.7176471  0.72156864 0.65882355]
  [0.7176471  0.72156864 0.65882355]]

 [[0.6261553  0.5908612  0.46929255]
  [0.5623564  0.5270623  0.40549365]
  [0.5856353  0.5503412  0.42877257]
  ...
  [0.7163428  0.7202644  0.6575193 ]
  [0.7176471  0.72156864 0.65882355]
  [0.7176471  0.72156864 0.65882355]]

 [[0.6261553  0.5908612  0.46929255]
  [0.5623564  0.5270623  0.40549365]
  [0.5856353  0.5503412  0.42877257]
  ...
  [0.7137255  0.7176471  0.654902  ]
  [0.7137255  0.7176471  0.654902  ]
  [0.7137255  0.7176471  0.654902  ]]

 ...

 [[0.63116914 0.48999268 0.39381915]
  [0.582721   0.43527347 0.34275374]
  [0.6325649  0.473082   0.37639308]
  ...
  [0.5398315  0.48624268 0.46140903]
  [0.51135015 0.4682129  0.4462831 ]
```

```
  [0.4990545  0.45594248 0.4402436 ]]

 [[0.631895   0.48700875 0.422937  ]
  [0.56681263 0.4231446  0.35037658]
  [0.5762563  0.4264816  0.33679175]
  ...
  [0.52072173 0.46607214 0.44176888]
  [0.49017024 0.44512686 0.43007594]
  [0.47559363 0.43704352 0.4202294 ]]

 [[0.48490584 0.33980778 0.31314287]
  [0.53454375 0.3925812  0.33533838]
  [0.57593566 0.44129798 0.33673722]
  ...
  [0.50559145 0.4441595  0.4219432 ]
  [0.48548847 0.43372205 0.41412282]
  [0.47372365 0.42745098 0.420653  ]]]


[[[0.578624   0.3903887  0.58646715]
  [0.6215754  0.43334007 0.62941855]
  [0.49759093 0.31645593 0.5089842 ]
  ...
  [0.26325837 0.17772657 0.12748888]
  [0.27628994 0.20457865 0.16480145]
  [0.3006913  0.23287258 0.20448603]]

 [[0.61618704 0.42795172 0.6240302 ]
  [0.6113049  0.4230696  0.619148  ]
  [0.7070106  0.5258756  0.7184039 ]
  ...
  [0.24945389 0.15663417 0.10872013]
  [0.24553862 0.16493326 0.12728953]
  [0.2408837  0.16906889 0.1292399 ]]

 [[0.55893517 0.37069988 0.5667783 ]
  [0.70517015 0.5169349  0.7130133 ]
  [0.5895018  0.40836677 0.60089505]
  ...
  [0.26999113 0.17982206 0.12494709]
  [0.26325724 0.17023039 0.12233356]
  [0.24769808 0.16199368 0.11158337]]
```

```
 ...

 [[0.8096003  0.78999245 0.76646304]
  [0.80336004 0.7837522  0.7602228 ]
  [0.852248   0.8255399  0.8020105 ]
  ...
  [0.26944718 0.39579257 0.4791193 ]
  [0.2515741  0.3776753  0.46831906]
  [0.3053066  0.43136695 0.5294062 ]]

 [[0.8003372  0.78734607 0.7605083 ]
  [0.78994405 0.7760055  0.7496415 ]
  [0.8452558  0.8185476  0.7890282 ]
  ...
  [0.2820229  0.41659603 0.5124823 ]
  [0.2575815  0.39761934 0.49775603]
  [0.32079074 0.46278837 0.5641359 ]]

 [[0.70227104 0.7173589  0.68114907]
  [0.71922666 0.7220174  0.68026304]
  [0.78168225 0.75594527 0.7164162 ]
  ...
  [0.28962258 0.4340369  0.53340405]
  [0.26090464 0.42537794 0.5296201 ]
  [0.30535218 0.47766483 0.5889433 ]]]


...


[[[0.09135355 0.04037315 0.01684374]
  [0.09135355 0.04037315 0.01684374]
  [0.09135355 0.04037315 0.01684374]
  ...
  [0.43309152 0.37034643 0.27230722]
  [0.43475857 0.37201348 0.28181738]
  [0.43475857 0.37201348 0.28181738]]

 [[0.11488296 0.06390256 0.04037315]
  [0.11488296 0.06390256 0.04037315]
  [0.11488296 0.06390256 0.04037315]
```

```
  ...
  [0.44429472 0.3815496  0.2835104 ]
  [0.4419798  0.3792347  0.28903863]
  [0.4419798  0.3792347  0.28903863]]

 [[0.10856008 0.05757969 0.02620713]
  [0.10856008 0.05757969 0.02620713]
  [0.10856008 0.05757969 0.02620713]
  ...
  [0.4503626  0.3876175  0.2895783 ]
  [0.43467218 0.37192708 0.281731  ]
  [0.43467218 0.37192708 0.281731  ]]

 ...

 [[0.99607843 0.99607843 0.99607843]
  [0.99607843 0.99607843 0.99607843]
  [0.99607843 0.99607843 0.99607843]
  ...
  [0.5019135  0.37517938 0.27594817]
  [0.5544967  0.45953274 0.37556463]
  [0.6041851  0.52183217 0.44732237]]

 [[0.99607843 0.99607843 0.99607843]
  [0.99607843 0.99607843 0.99607843]
  [0.99607843 0.99607843 0.99607843]
  ...
  [0.44124848 0.32916316 0.24065153]
  [0.47073197 0.38522625 0.30441093]
  [0.49124923 0.4088963  0.3343865 ]]

 [[0.99607843 0.99607843 0.99607843]
  [0.99607843 0.99607843 0.99607843]
  [0.99607843 0.99607843 0.99607843]
  ...
  [0.5053434  0.41174996 0.33789068]
  [0.5621985  0.47848442 0.40125233]
  [0.5303601  0.4480071  0.3734973 ]]]


[[[0.13912977 0.10562367 0.0667535 ]
  [0.19454591 0.14356552 0.11562354]
```

```
 [0.28072995 0.21006395 0.19445542]
 ...
 [0.41174102 0.38429004 0.4156626 ]
 [0.4103222  0.3828712  0.41424376]
 [0.4103222  0.3828712  0.41424376]]

[[0.12472143 0.08942731 0.06693444]
 [0.19960491 0.14498574 0.13237268]
 [0.28388956 0.21319768 0.21038355]
 ...
 [0.42709693 0.39964595 0.4310185 ]
 [0.43633068 0.4088797  0.44025224]
 [0.43633068 0.4088797  0.44025224]]

[[0.13572055 0.09537877 0.10048462]
 [0.18656045 0.13110477 0.13188396]
 [0.28537756 0.21187977 0.21756208]
 ...
 [0.44881964 0.42136866 0.4527412 ]
 [0.4459618  0.41851082 0.44988337]
 [0.4459618  0.41851082 0.44988337]]

...

[[0.77188945 0.7091443  0.71306586]
 [0.77057016 0.70782506 0.71174663]
 [0.7653179  0.7025728  0.7064944 ]
 ...
 [0.17687763 0.13766195 0.1807992 ]
 [0.10840502 0.08249165 0.11232659]
 [0.06310218 0.04349433 0.06702375]]

[[0.7580204  0.6952753  0.6991969 ]
 [0.74622977 0.6834847  0.68740624]
 [0.74075425 0.67800915 0.6819307 ]
 ...
 [0.18393657 0.14472088 0.18785813]
 [0.14661805 0.11916707 0.15053962]
 [0.09570695 0.06825598 0.09962852]]

[[0.7560681  0.693323   0.6972446 ]
 [0.7458668  0.6831217  0.68704325]
```

```
  [0.74502033 0.68227524 0.6861968 ]
  ...
  [0.14927931 0.11006362 0.15320088]
  [0.14125402 0.10203833 0.14517559]
  [0.09570695 0.05649127 0.09962852]]]


[[[0.7417522  0.76136005 0.73783064]
  [0.7422303  0.76183814 0.7383087 ]
  [0.73213565 0.7517435  0.7282141 ]
  ...
  [0.6580224  0.7168459  0.7442969 ]
  [0.6684304  0.71516097 0.7557586 ]
  [0.27425608 0.29642072 0.3681203 ]]

 [[0.715781   0.7353888  0.7118594 ]
  [0.6900411  0.70964897 0.68611956]
  [0.71768236 0.7372902  0.7137608 ]
  ...
  [0.70507234 0.75622547 0.7875116 ]
  [0.6110452  0.66117847 0.69890904]
  [0.29993805 0.346565   0.39711347]]

 [[0.65902    0.67862785 0.65509844]
  [0.6552463  0.67485416 0.65132475]
  [0.67574084 0.6953487  0.67181927]
  ...
  [0.7525278  0.7985673  0.8171903 ]
  [0.5073535  0.56063026 0.5878567 ]
  [0.3488911  0.4204986  0.4490841 ]]

 ...

 [[0.01176471 0.01176471 0.01176471]
  [0.01176471 0.01176471 0.01176471]
  [0.01176471 0.01176471 0.01176471]
  ...
  [0.6804633  0.6686986  0.6334045 ]
  [0.6778434  0.66846263 0.63237387]
  [0.68695366 0.68695366 0.647738  ]]

 [[0.01176471 0.01176471 0.01176471]
```

```
  [0.01176471 0.01176471 0.01176471]
  [0.01176471 0.01176471 0.01176471]
  ...
  [0.66664946 0.65488476 0.6195906 ]
  [0.6588749  0.6494942  0.6134054 ]
  [0.66423076 0.66423076 0.6250151 ]]

 [[0.01176471 0.01176471 0.01176471]
  [0.01176471 0.01176471 0.01176471]
  [0.01176471 0.01176471 0.01176471]
  ...
  [0.66664946 0.65488476 0.6195906 ]
  [0.6611558  0.65177506 0.6156863 ]
  [0.6603956  0.6603956  0.62117994]]]]
```

In [ ]:
```
classes=['cat', 'dog']
```

In [ ]:
```
# A color bar is more meaningful when displaying data such as grayscale images,
# heatmaps, or other data visualizations where colors represent a range of
# scalar values. For RGB images, the color bar does not provide additional
# useful information and is generally not included.

# So we do not need plt.colorbar().

def show_img(image, label):
  plt.figure(figsize=(20, 2))
  plt.imshow(image)
  plt.xlabel(classes[label])
  plt.colorbar()
  plt.show()
```

In [ ]:
```
show_img(sample_elements[0][0].numpy()[5], sample_elements[0][1].numpy()[5])
```

In [ ]:

```
# Building an Adaptation of Alexnet Model:

model = Sequential()

# For Conv2D():

# kernel_size is the filter size(k). Use odd number of filters in convolution.
# so that when padding is 'same' the value p comes out as a whole number.

# padding='valid' by default this means no padding.
# padding='same' means that the padding is automatically calculated to ensure
# that the output (feature map) has the same spatial dimensions as the input
# image using the formula p = (k-1)/2.

# strides=(1, 1) by default.

# input_shape=(batchSize, height, width, channels)
# Specifying the batchSize here in input_shape won't have any effect. The
# recommended place to specify the batchSize is the model.fit() method.

# Conv Layer - 1
model.add(Conv2D(filters=96, kernel_size=(11,11), strides=(4,4),
                 activation='relu', input_shape=(227,227,3)))

model.add(BatchNormalization())

# For MaxPooling2D():
```

```
# pool_size=(2, 2) by default
# strides=(2, 2) by default.
# output_shape = math.floor((input_shape - pool_size) / strides) + 1
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2)))

# Conv Layer - 2
model.add(Conv2D(filters=256, kernel_size=(5,5),
                 padding='same', activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2)))

# Conv Layer - 3
model.add(Conv2D(filters=384, kernel_size=(3,3),
                 padding='same', activation='relu'))

# Conv Layer - 4
model.add(Conv2D(filters=384, kernel_size=(3,3),
                 padding='same', activation='relu'))

# Conv Layer - 5
model.add(Conv2D(filters=256, kernel_size=(3,3),
                 padding='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2)))

# Flatten layer in Keras converts a multi-dimensional array into a 1D array.
model.add(Flatten())

model.add(Dense(units=256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(units=256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(units=1, activation='sigmoid'))

# Model compilation
```

```
# keras.metrics.SparseCategoricalAccuracy(): Calculates how often predictions
# match integer labels.
# When loss='sparse_categorical_crossentropy', 'accuracy' metric works same
# as keras.metrics.SparseCategoricalAccuracy()
model.compile(optimizer=keras.optimizers.SGD(weight_decay=0.0005, momentum=0.9),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 55, 55, 96)        34944

 batch_normalization_4 (Bat  (None, 55, 55, 96)        384
 chNormalization)

 max_pooling2d_6 (MaxPoolin  (None, 27, 27, 96)        0
 g2D)

 conv2d_11 (Conv2D)          (None, 27, 27, 256)       614656

 batch_normalization_5 (Bat  (None, 27, 27, 256)       1024
 chNormalization)

 max_pooling2d_7 (MaxPoolin  (None, 13, 13, 256)       0
 g2D)

 conv2d_12 (Conv2D)          (None, 13, 13, 384)       885120

 conv2d_13 (Conv2D)          (None, 13, 13, 384)       1327488

 conv2d_14 (Conv2D)          (None, 13, 13, 256)       884992

 max_pooling2d_8 (MaxPoolin  (None, 6, 6, 256)         0
 g2D)

 flatten_2 (Flatten)         (None, 9216)              0

 dense_6 (Dense)             (None, 256)               2359552

 dropout_4 (Dropout)         (None, 256)               0

 dense_7 (Dense)             (None, 256)               65792

 dropout_5 (Dropout)         (None, 256)               0

 dense_8 (Dense)             (None, 1)                 257

=================================================================
```

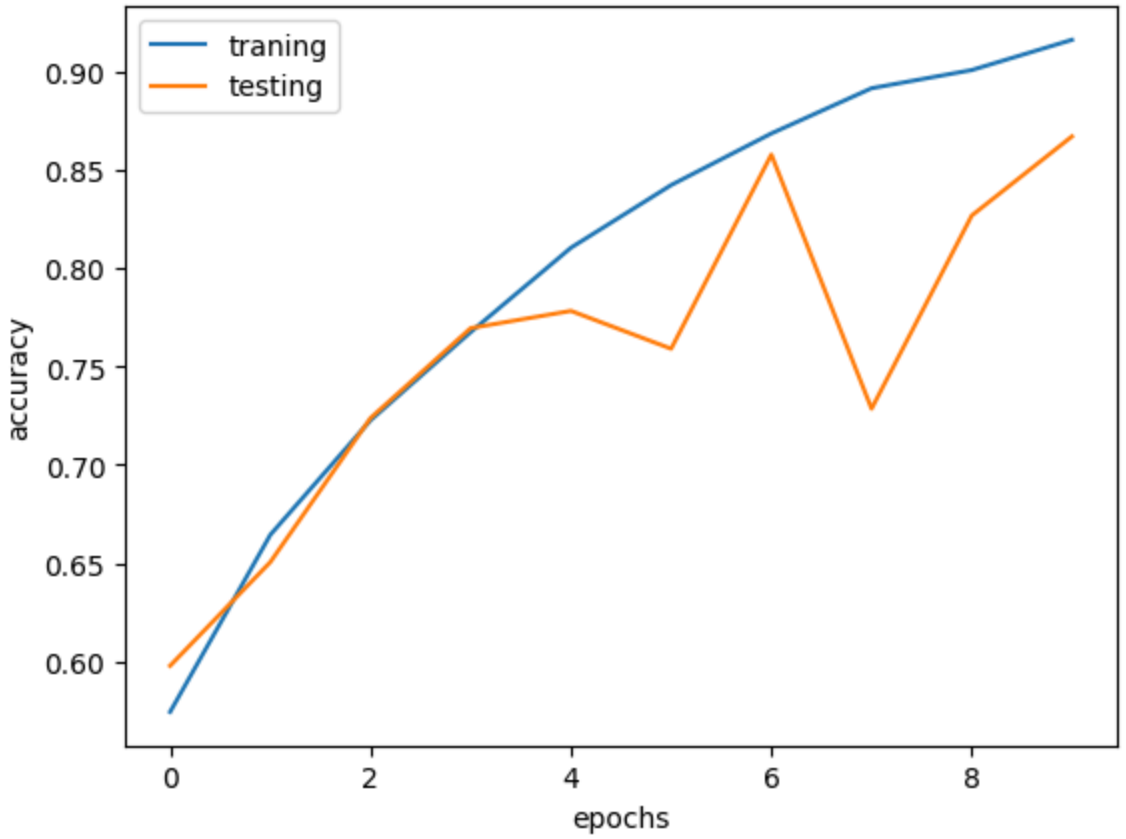
```
Total params: 6174209 (23.55 MB)
Trainable params: 6173505 (23.55 MB)
Non-trainable params: 704 (2.75 KB)
_________________________________________________________________
```

In [ ]:
```
model_history = model.fit(train_dataset, epochs=10,
                          validation_data=validation_dataset)
```

```
Epoch 1/10
500/500 [==============================] - 36s 66ms/step - loss: 0.6806 - accuracy: 0.5743 - val_loss: 0.6510 - val_a
ccuracy: 0.5978
Epoch 2/10
500/500 [==============================] - 33s 65ms/step - loss: 0.6154 - accuracy: 0.6644 - val_loss: 0.6178 - val_a
ccuracy: 0.6505
Epoch 3/10
500/500 [==============================] - 39s 77ms/step - loss: 0.5504 - accuracy: 0.7226 - val_loss: 0.5616 - val_a
ccuracy: 0.7237
Epoch 4/10
500/500 [==============================] - 39s 77ms/step - loss: 0.4866 - accuracy: 0.7675 - val_loss: 0.4786 - val_a
ccuracy: 0.7695
Epoch 5/10
500/500 [==============================] - 33s 66ms/step - loss: 0.4167 - accuracy: 0.8104 - val_loss: 0.4725 - val_a
ccuracy: 0.7782
Epoch 6/10
500/500 [==============================] - 34s 67ms/step - loss: 0.3545 - accuracy: 0.8422 - val_loss: 0.4914 - val_a
ccuracy: 0.7590
Epoch 7/10
500/500 [==============================] - 33s 66ms/step - loss: 0.3023 - accuracy: 0.8685 - val_loss: 0.3423 - val_a
ccuracy: 0.8577
Epoch 8/10
500/500 [==============================] - 32s 64ms/step - loss: 0.2631 - accuracy: 0.8915 - val_loss: 0.6250 - val_a
ccuracy: 0.7285
Epoch 9/10
500/500 [==============================] - 34s 67ms/step - loss: 0.2338 - accuracy: 0.9007 - val_loss: 0.4491 - val_a
ccuracy: 0.8267
Epoch 10/10
500/500 [==============================] - 33s 65ms/step - loss: 0.2034 - accuracy: 0.9161 - val_loss: 0.3384 - val_a
ccuracy: 0.8670
```

In [ ]:
```
plt.figure()
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
```

```
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['traning', 'testing'])
plt.show()
```



In [ ]:
```
model.evaluate(test_dataset)
```

```
157/157 [==============================] - 7s 43ms/step - loss: 0.3159 - accuracy: 0.8706
```

Out[ ]:
```
[0.3158647418022156, 0.8705999851226807]
```

In [ ]:
```
y_test_pred = model.predict(test_dataset)
```

```
157/157 [==============================] - 8s 51ms/step
```

In [ ]:
```
y_test_pred
```

Out[ ]:
```
array([[1.1369628e-01],
       [9.9094957e-01],
       [9.5372963e-01],
       ...,
       [8.9531073e-07],
       [7.2885710e-01],
       [9.9999464e-01]], dtype=float32)
```

In [ ]:
```
y_test_pred.shape
```

Out[ ]:
```
(5000, 1)
```

In [ ]:
```
X_test_list = []
y_test_list = []
for images_batch in test_dataset:
  X_test_list.append(images_batch[0].numpy())
  y_test_list.append(images_batch[1].numpy())
```

In [ ]:
```
print(len(X_test_list))
print(len(y_test_list))
```

```
157
157
```

In [ ]:
```
X_test = np.concatenate(X_test_list)
y_test = np.concatenate(y_test_list)
```

In [ ]:
```
print(X_test.shape)
print(y_test.shape)
```

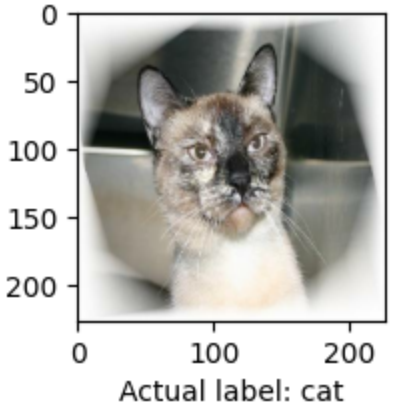
```
(5000, 227, 227, 3)
(5000,)
```

In [ ]:
```
# First 10 data points predictions.
y_pred_class_test = np.where(y_test >= 0.5, 1, 0)
print(f'Actual labels: {y_test[:10]}')
print(f'Predicted labels: {y_pred_class_test[:10]}')
```

```
Actual labels: [0 1 0 0 0 0 0 0 0 0]
Predicted labels: [0 1 0 0 0 0 0 0 0 0]
```

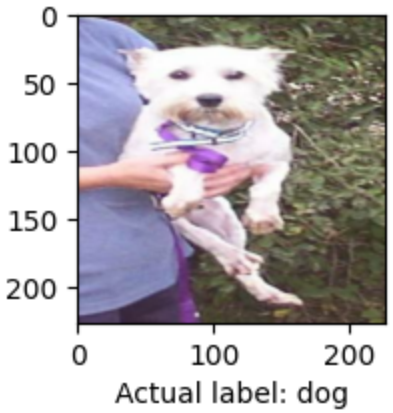
In [ ]:
```
def act_to_pred(val):
    plt.figure(figsize=(18, 2))
    plt.imshow(X_test[val, :])
    plt.xlabel(f'Actual label: {classes[y_test[val]]}')
    plt.show()
    print('The predicted image label is:',classes[y_pred_class_test[val]])
```

In [ ]:
```
act_to_pred(0)
```



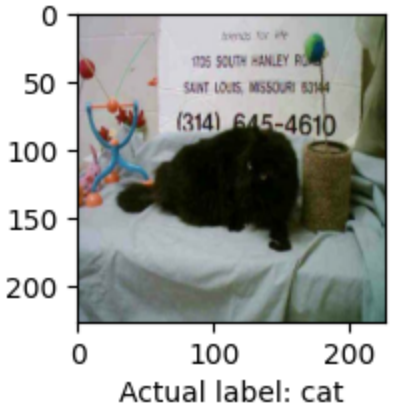

```
The predicted image label is: cat
```

In [ ]:
```
act_to_pred(3037)
```



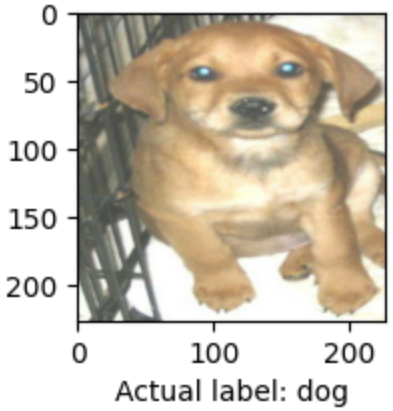

```
The predicted image label is: dog
```

In [ ]:
```
act_to_pred(2001)
```




```
The predicted image label is: cat
```

In [ ]:
```
act_to_pred(955)
```

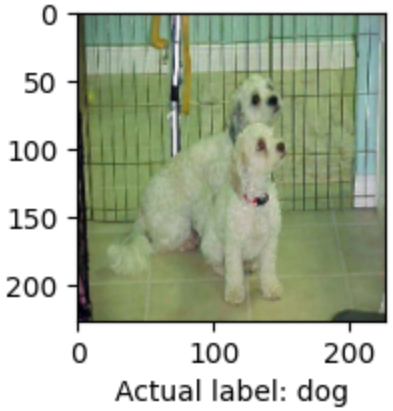



```
The predicted image label is: dog
```

In [ ]:
```
act_to_pred(30)
```

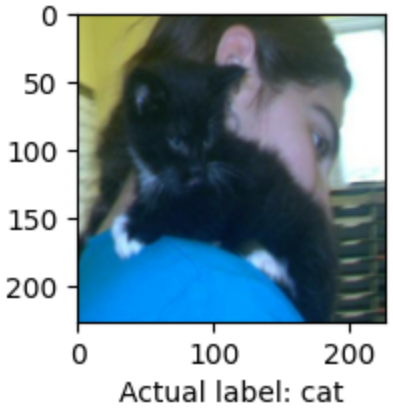

```
The predicted image label is: dog
```

In [ ]:
```
act_to_pred(1)
```




```
The predicted image label is: dog
```

In [ ]:
```
act_to_pred(4)
```




```
The predicted image label is: cat
```

Conclusion:

The **AlexNet model** to classify a given input image as a Cat or Dog has a **train accuracy of 91.6%**, **validation accuracy of 86.7%** and **test accuracy of 87.06%**.