**Practical Assignment**

**Objective: - Object Detection with OpenImages**

Open Images is a new dataset first released in 2016 that contains ~9 million images – which is fewer than ImageNet. What makes it stand out is that these images are mostly of complex scenes that span thousands of classes of objects. Moreover, ~2 million of these images are hand-annotated with bounding boxes making Open Images by far the largest existing dataset with object location annotations. In this subset of images, there are ~15.4 million bounding boxes of 600 classes of object.

**Dataset Link: -** Dataset is pretty big. SO we do not want to train it completely. So please extract any 10 classes images and annotations that you like. Then train it.

Link :-<https://github.com/cvdfoundation/open-images-dataset>

**Task: -** Create a Web Application using Flask. Use the end user should be able to upload an image and get results with the prediction score. Use any CNN architecture launched after 2017.

**Deployment: -** Any Free Platform(Try to look out for free options.)

**Assignment Submission: -** Only submit the hosted app link. OR GitHub Link

# python3

# coding=utf-8

# Copyright 2020 The Google Research Authors.

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

"""Open Images image downloader.

This script downloads a subset of Open Images images, given a list of image ids.

Typical uses of this tool might be downloading images:

- That contain a certain category.

- That have been annotated with certain types of annotations (e.g. Localized

Narratives, Exhaustively annotated people, etc.)

The input file IMAGE\_LIST should be a text file containing one image per line

with the format <SPLIT>/<IMAGE\_ID>, where <SPLIT> is either "train", "test",

"validation", or "challenge2018"; and <IMAGE\_ID> is the image ID that uniquely

identifies the image in Open Images. A sample file could be:

train/f9e0434389a1d4dd

train/1a007563ebc18664

test/ea8bfd4e765304db

"""

import argparse

from concurrent import futures

import os

import re

import sys

import boto3

import botocore

import tqdm

BUCKET\_NAME = 'open-images-dataset'

REGEX = r'(test|train|validation|challenge2018)/([a-fA-F0-9]\*)'

def check\_and\_homogenize\_one\_image(image):

split, image\_id = re.match(REGEX, image).groups()

yield split, image\_id

def check\_and\_homogenize\_image\_list(image\_list):

for line\_number, image in enumerate(image\_list):

try:

yield from check\_and\_homogenize\_one\_image(image)

except (ValueError, AttributeError):

raise ValueError(

f'ERROR in line {line\_number} of the image list. The following image '

f'string is not recognized: "{image}".')

def read\_image\_list\_file(image\_list\_file):

with open(image\_list\_file, 'r') as f:

for line in f:

yield line.strip().replace('.jpg', '')

def download\_one\_image(bucket, split, image\_id, download\_folder):

try:

bucket.download\_file(f'{split}/{image\_id}.jpg',

os.path.join(download\_folder, f'{image\_id}.jpg'))

except botocore.exceptions.ClientError as exception:

sys.exit(

f'ERROR when downloading image `{split}/{image\_id}`: {str(exception)}')

def download\_all\_images(args):

"""Downloads all images specified in the input file."""

bucket = boto3.resource(

's3', config=botocore.config.Config(

signature\_version=botocore.UNSIGNED)).Bucket(BUCKET\_NAME)

download\_folder = args['download\_folder'] or os.getcwd()

if not os.path.exists(download\_folder):

os.makedirs(download\_folder)

try:

image\_list = list(

check\_and\_homogenize\_image\_list(

read\_image\_list\_file(args['image\_list'])))

except ValueError as exception:

sys.exit(exception)

progress\_bar = tqdm.tqdm(

total=len(image\_list), desc='Downloading images', leave=True)

with futures.ThreadPoolExecutor(

max\_workers=args['num\_processes']) as executor:

all\_futures = [

executor.submit(download\_one\_image, bucket, split, image\_id,

download\_folder) for (split, image\_id) in image\_list

]

for future in futures.as\_completed(all\_futures):

future.result()

progress\_bar.update(1)

progress\_bar.close()

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser(

description=\_\_doc\_\_, formatter\_class=argparse.RawDescriptionHelpFormatter)

parser.add\_argument(

'image\_list',

type=str,

default=None,

help=('Filename that contains the split + image IDs of the images to '

'download. Check the document'))

parser.add\_argument(

'--num\_processes',

type=int,

default=5,

help='Number of parallel processes to use (default is 5).')

parser.add\_argument(

'--download\_folder',

type=str,

default=None,

help='Folder where to download the images.')

download\_all\_images(vars(parser.parse\_args()))