|  |  |
| --- | --- |
|  | import numpy as np |
|  | import os |
|  | import six.moves.urllib as urllib |
|  | import sys |
|  | import tarfile |
|  | import tensorflow as tf |
|  | import zipfile |
|  | import time |
|  | from imutils.video import WebcamVideoStream |
|  | from imutils.video import FPS |
|  | import imutils |
|  |  |
|  | from collections import defaultdict |
|  | from io import StringIO |
|  | from PIL import Image |
|  |  |
|  | if tf.\_\_version\_\_ < '1.4.0': |
|  | raise ImportError('Please upgrade your tensorflow installation to v1.4.\* or later!') |
|  | # This is needed to display the images. |
|  |  |
|  |  |
|  | # This is needed since the notebook is stored in the object\_detection folder. |
|  | sys.path.append("..") |
|  |  |
|  |  |
|  | from utils import label\_map\_util |
|  |  |
|  | from utils import visualization\_utils as vis\_util |
|  |  |
|  | # What model to download. |
|  | MODEL\_NAME = 'training' |
|  |  |
|  | # Path to frozen detection graph. This is the actual model that is used for the object detection. |
|  | PATH\_TO\_CKPT = MODEL\_NAME + '/frozen\_inference\_graph.pb' |
|  |  |
|  | # List of the strings that is used to add correct label for each box. |
|  | #PATH\_TO\_LABELS = os.path.join('training', 'License-Plate-Detection.pbtxt') |
|  |  |
|  | NUM\_CLASSES = 14 |
|  |  |
|  |  |
|  | '''tar\_file = tarfile.open(MODEL\_FILE) |
|  | for file in tar\_file.getmembers(): |
|  | file\_name = os.path.basename(file.name) |
|  | if 'frozen\_inference\_graph.pb' in file\_name: |
|  | tar\_file.extract(file, os.getcwd())''' |
|  |  |
|  | detection\_graph = tf.Graph() |
|  | with detection\_graph.as\_default(): |
|  | od\_graph\_def = tf.GraphDef() |
|  | with tf.gfile.GFile(PATH\_TO\_CKPT, 'rb') as fid: |
|  | serialized\_graph = fid.read() |
|  | od\_graph\_def.ParseFromString(serialized\_graph) |
|  | tf.import\_graph\_def(od\_graph\_def, name='') |
|  |  |
|  |  |
|  | category\_index={1: {'id': 1, 'name': u'person'}, |
|  | 2: {'id': 2, 'name': u'bicycle'}, |
|  | 3: {'id': 3, 'name': u'car'}, |
|  | 4: {'id': 4, 'name': u'motorcycle'}, |
|  | 5: {'id': 5, 'name': u'airplane'}, |
|  | 6: {'id': 6, 'name': u'bus'}, |
|  | 7: {'id': 7, 'name': u'train'}, |
|  | 8: {'id': 8, 'name': u'truck'}, |
|  | 9: {'id': 9, 'name': u'boat'}, |
|  | 10: {'id': 10, 'name': u'traffic light'}, |
|  | 11: {'id': 11, 'name': u'fire hydrant'}, |
|  | 13: {'id': 13, 'name': u'stop sign'}, |
|  | 14: {'id': 14, 'name': u'parking meter'}} |
|  |  |
|  | import cv2 |
|  | cap=cv2.VideoCapture('video.mp4') # 0 stands for very first webcam attach |
|  | filename="testoutput.avi" |
|  | codec=cv2.VideoWriter\_fourcc('m','p','4','v')#fourcc stands for four character code |
|  | framerate=30 |
|  | resolution=(640,480) |
|  |  |
|  | VideoFileOutput=cv2.VideoWriter(filename,codec,framerate, resolution) |
|  | vs = WebcamVideoStream(src='test.mp4').start() |
|  |  |
|  | with detection\_graph.as\_default(): |
|  | with tf.Session(graph=detection\_graph) as sess: |
|  |  |
|  | ret=True |
|  | start=time.time() |
|  | c=0 |
|  |  |
|  |  |
|  |  |
|  |  |
|  | '''fps = FPS().start() |
|  |  |
|  | # loop over some frames...this time using the threaded stream |
|  | while True: |
|  | # grab the frame from the threaded video stream and resize it |
|  | # to have a maximum width of 400 pixels |
|  | frame = vs.read() |
|  | frame = imutils.resize(frame, width=400) |
|  |  |
|  | cv2.imshow("Frame", frame) |
|  | key = cv2.waitKey(1) & 0xFF |
|  |  |
|  | # update the FPS counter |
|  | fps.update() |
|  |  |
|  | # stop the timer and display FPS information |
|  | fps.stop() |
|  | ''' |
|  |  |
|  |  |
|  | while (ret): |
|  |  |
|  | r,image\_np=cap.read() |
|  | #image\_np = imutils.resize(image\_np, width=400) |
|  | c=c+1 |
|  | # Definite input and output Tensors for detection\_graph |
|  | image\_tensor = detection\_graph.get\_tensor\_by\_name('image\_tensor:0') |
|  | # Each box represents a part of the image where a particular object was detected. |
|  | detection\_boxes = detection\_graph.get\_tensor\_by\_name('detection\_boxes:0') |
|  | # Each score represent how level of confidence for each of the objects. |
|  | # Score is shown on the result image, together with the class label. |
|  | detection\_scores = detection\_graph.get\_tensor\_by\_name('detection\_scores:0') |
|  | detection\_classes = detection\_graph.get\_tensor\_by\_name('detection\_classes:0') |
|  | num\_detections = detection\_graph.get\_tensor\_by\_name('num\_detections:0') |
|  |  |
|  | # Expand dimensions since the model expects images to have shape: [1, None, None, 3] |
|  | image\_np\_expanded = np.expand\_dims(image\_np, axis=0) |
|  | # Actual detection. |
|  | (boxes, scores, classes, num) = sess.run( |
|  | [detection\_boxes, detection\_scores, detection\_classes, num\_detections], |
|  | feed\_dict={image\_tensor: image\_np\_expanded}) |
|  | # Visualization of the results of a detection. |
|  | vis\_util.visualize\_boxes\_and\_labels\_on\_image\_array( |
|  | image\_np, |
|  | np.squeeze(boxes), |
|  | np.squeeze(classes).astype(np.int32), |
|  | np.squeeze(scores), |
|  | category\_index, |
|  | use\_normalized\_coordinates=True, |
|  | line\_thickness=8) |
|  |  |
|  | #VideoFileOutput.write(image\_np) |
|  | cv2.imshow('live\_detection',image\_np) |
|  | if cv2.waitKey(25) & 0xFF==ord('q'): |
|  | elapsed=time.time()-start |
|  | print('Run Time = ',elapsed) |
|  | print('fps = ',c/elapsed) |
|  | break |
|  | cv2.destroyAllWindows() |
|  | cap.release() |