Object Detection in Live Cam using Tensorflow Object Detection API

Import Useful Libraries

In [1]:

```
import numpy as np
import os
import six.moves.urllib as urllib
import tarfile
import tensorflow as tf

import cv2
cap = cv2.VideoCapture(0)

from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util
```

Model Preparation

In [2]:

```
# # Any model exported using the `export_inference_graph.py` tool can be loaded here simply by changing `PATH_TO_FROZEN_GRAPH`
# # to point to a new .pb file. By default we use an "SSD with Mobilenet" model here.

# # See https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md for a list of
# # other models that can be run out-of-the-box with varying speeds and accuracies.

# # What model to download.
# MODEL_NAME = 'ssd_mobilenet_v1_coco_2017_11_17'
# MODEL_FILE = MODEL_NAME + '.tar.gz'
# DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# # Path to frozen detection graph. This is the actual model that is used for the object detection.
# PATH_TO_FROZEN_GRAPH = 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('data', 'mscoco_label_map.pbtxt')

# # Download Model
# opener = urllib.request.URLopener()
# opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
# 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())
```

Load a (frozen) Tensorflow model into memory

In [3]:

```
PATH_TO_LABELS = "C:/Users/HP/AppData/Roaming/Python/Python37/site-packages/object_detection/data/mscoco_label_map.pbtxt"
PATH_TO_FROZEN_GRAPH = "E:/SOFTWARES/Object Detection/ssd_mobilenet_v1_coco_2017_11_17/frozen_inference_graph.pb"

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
```

Loading label map

Label maps map indices to category names, so that when our convolution network predicts `5`, we know that this corresponds to `airplane`. Here we use internal utility functions, but anything that returns a dictionary mapping integers to appropriate string labels would be fine.

In [4]:

```
NUM_CLASSES = 90

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
```

To Start Live Cam Object Detection

In [5]:

```
with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        while True:
            ret, image_np = cap.read()
            # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
            image_np_expanded = np.expand_dims(image_np, axis=0)
            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.
            boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
            # Each score represent how level of confidence for each of the objects.
            scores = detection_graph.get_tensor_by_name('detection_scores:0')
            classes = detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections = detection_graph.get_tensor_by_name('num_detections:0')
            # Actual detection.
            (boxes, scores, classes, num_detections) = sess.run([boxes, scores, 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)
            cv2.imshow('Object Detection', cv2.resize(image_np, (800, 600)))
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break

cap.release()
cv2.destroyAllWindows()
```