UMKC

School of Computing
and Engineering

Python and Deep Learning

Project

TensorFlow Object Detection

Team: Blue and Gold

Team Members:

1. Archana Reddy Basani, Class ID: 5
2. Sree Prathyusha Nomula, Class ID: 21
3. Devi Vara Prasad Kada, Class ID:13
4. Sushmanth Makkena, Classs ID:16

Introduction:

TensorFlow object detection model is an open source framework which is built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. The purpose of this project is to create precise machine learning models which has the ability of localizing and identifying multiple objects in a single image.

Objectives:

- Identifying and detection multiple objects in a single image and differentiating between them.
- Giving the accuracy of the objects in a webcam or video or an image.
- Generating Video stream using the webcam.

Requirements and Installation:

- First, we need to install the TensorFlow using PIP command.
- We need to download TensorFlow Object Detection API and Protocols from TensorFlow GitHub.
- The Version of the Protocol Should be 3.4 i.e. Protoc34.
- Import the cv2 file for the video capturing and object detection of the video.

Configuration:

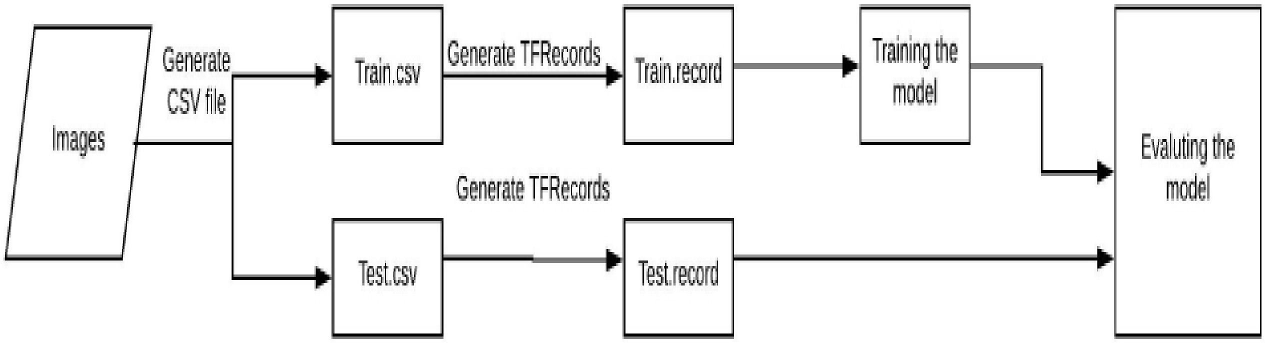
- Python 3.6.4
- Protoc 34
- TensorFlow library

TensorFlow Overview:

This application has the following components:

- TensorFlow: It runs on multiple computers to allocate the training workloads.
- Object Detection: Used to construct, train and deploy object detection models.
- Pre-trained object detection models: The object detection API gives the pre-trained object detection models. In this case, users are not required to train models from scratch.

Flow Diagram:



Work Flow:

Step 1: Creating the csv files from images.

The objective is to label the image and generate train.csv and test.csv files.

Step 2: Generating TF records.

In this case, TensorFlow object detection doesn't take csv files as an input, but it needs record files to train the model.

Step 3: Training the Model.

In this, we will copy image directories to object detection directories.

Step 4: Evaluating the model.

The last step is to evaluate the training model saved in training directory.

Input/output Screen shots:

Task: Imports

The Objects detection requires some of the libraries

1.Tensor Flow Library

```
pip install tensorflow
```

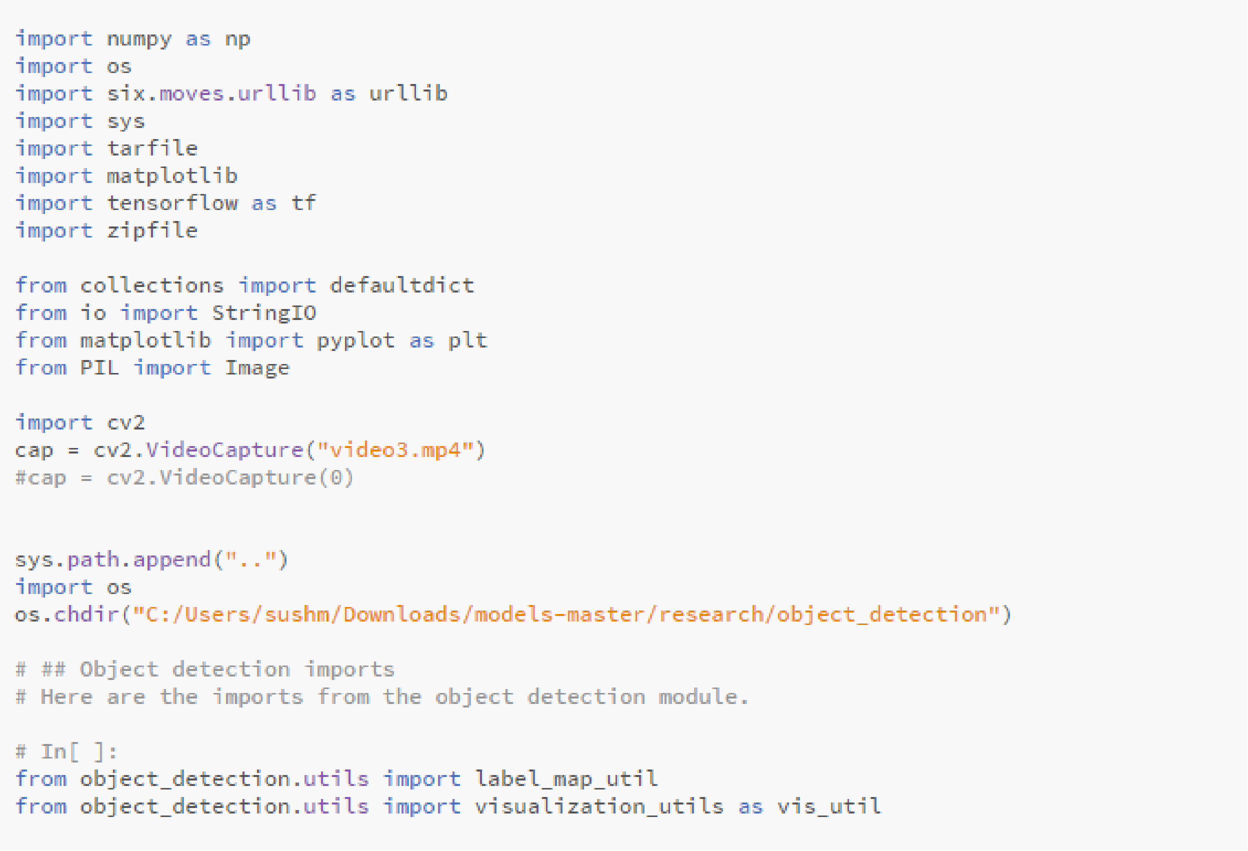
2. Dependencies

```
pip install --user Cython
pip install --user contextlib2
pip install --user pillow
pip install --user lxml
pip install --user jupyter
pip install --user matplotlib
```

3. Adding Libraries to pythonpath

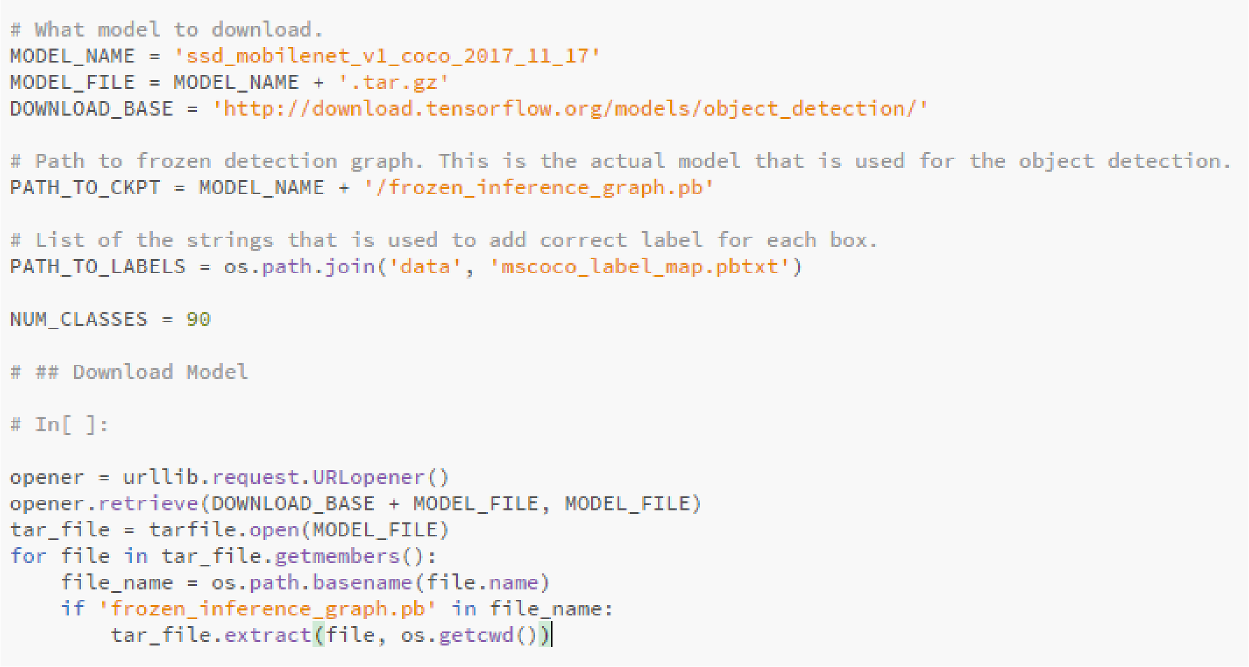
```
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

Implementation and Code Snippet:



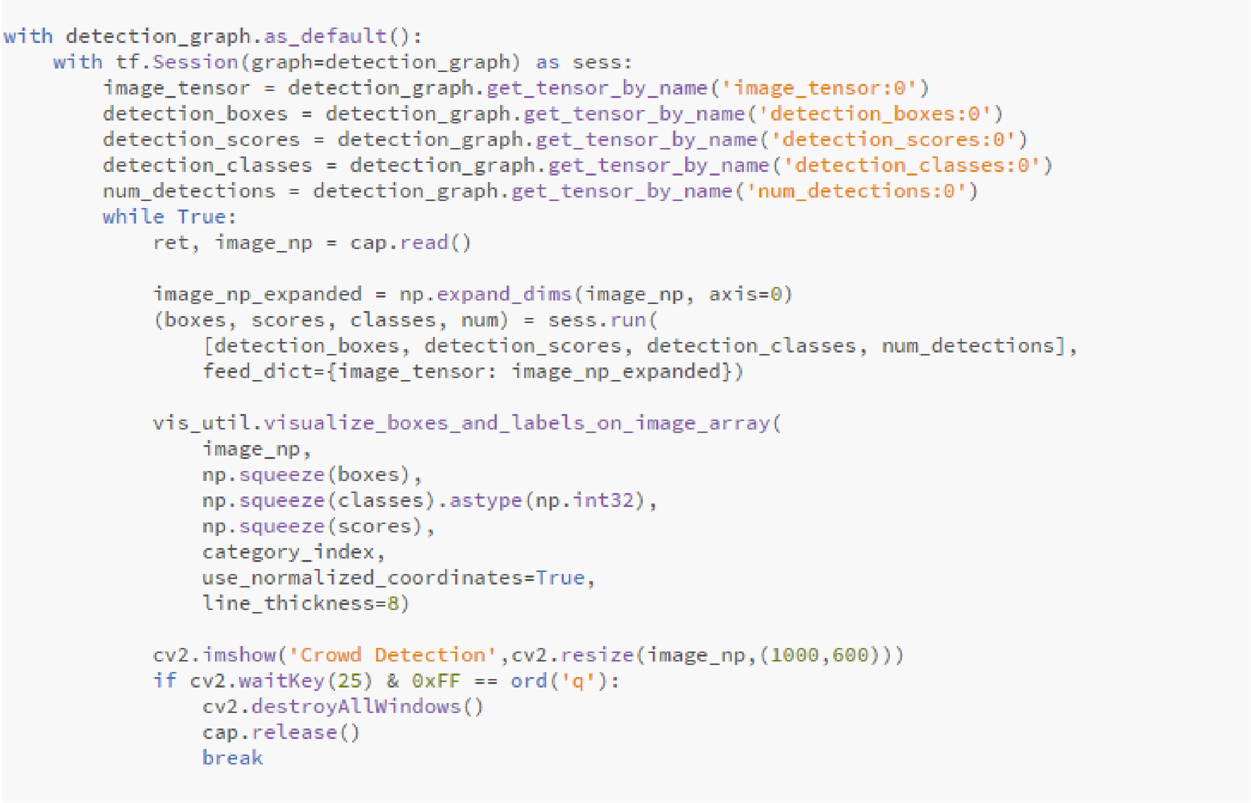
```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import matplotlib
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

import cv2
cap = cv2.VideoCapture("video3.mp4")
#cap = cv2.VideoCapture(0)


sys.path.append("..")
import os
os.chdir("C:/Users/sushm/Downloads/models-master/research/object_detection")

# ## Object detection imports
# Here are the imports from the object detection module.

# In[ ]:
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util
```

Task 2: Downloading the models



```
# 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_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('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90

# ## Download Model

# In[ ]:

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())
```

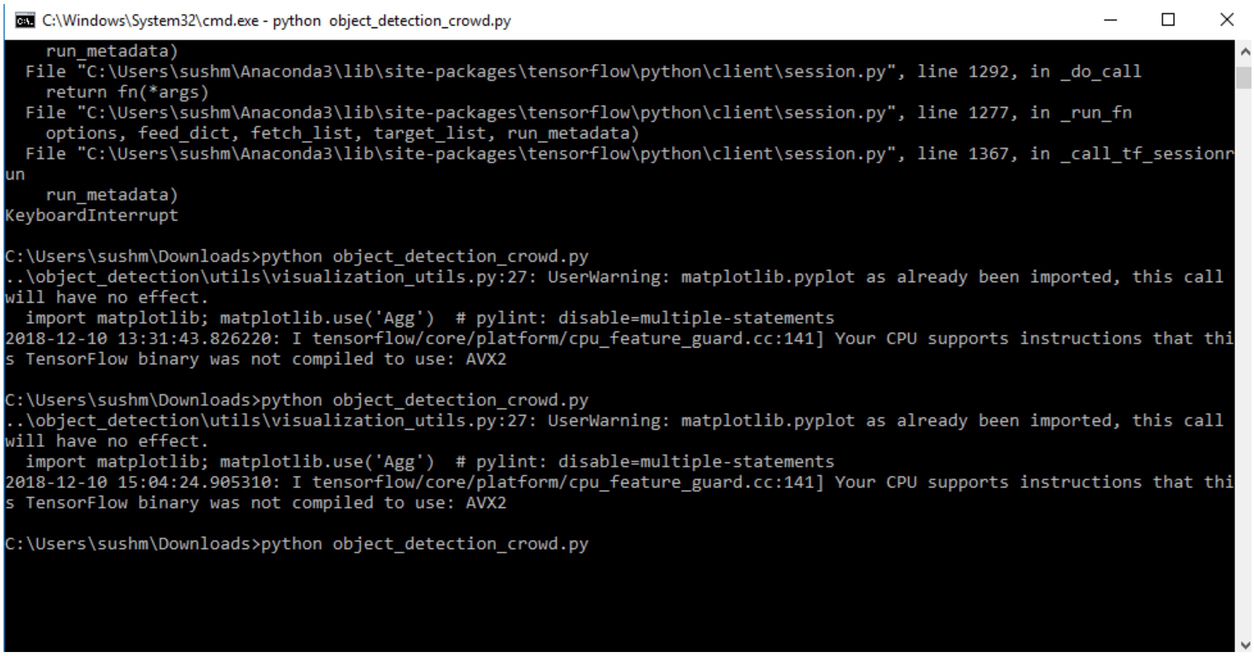
Task 3: Detection Graph



```
with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
        detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
        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')
        while True:
            ret, image_np = cap.read()

            image_np_expanded = np.expand_dims(image_np, axis=0)
            (boxes, scores, classes, num) = sess.run(
                [detection_boxes, detection_scores, detection_classes, num_detections],
                feed_dict={image_tensor: image_np_expanded})

            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('Crowd Detection',cv2.resize(image_np,(1000,600)))
            if cv2.waitKey(25) & 0xFF == ord('q'):
                cv2.destroyAllWindows()
                cap.release()
                break
```

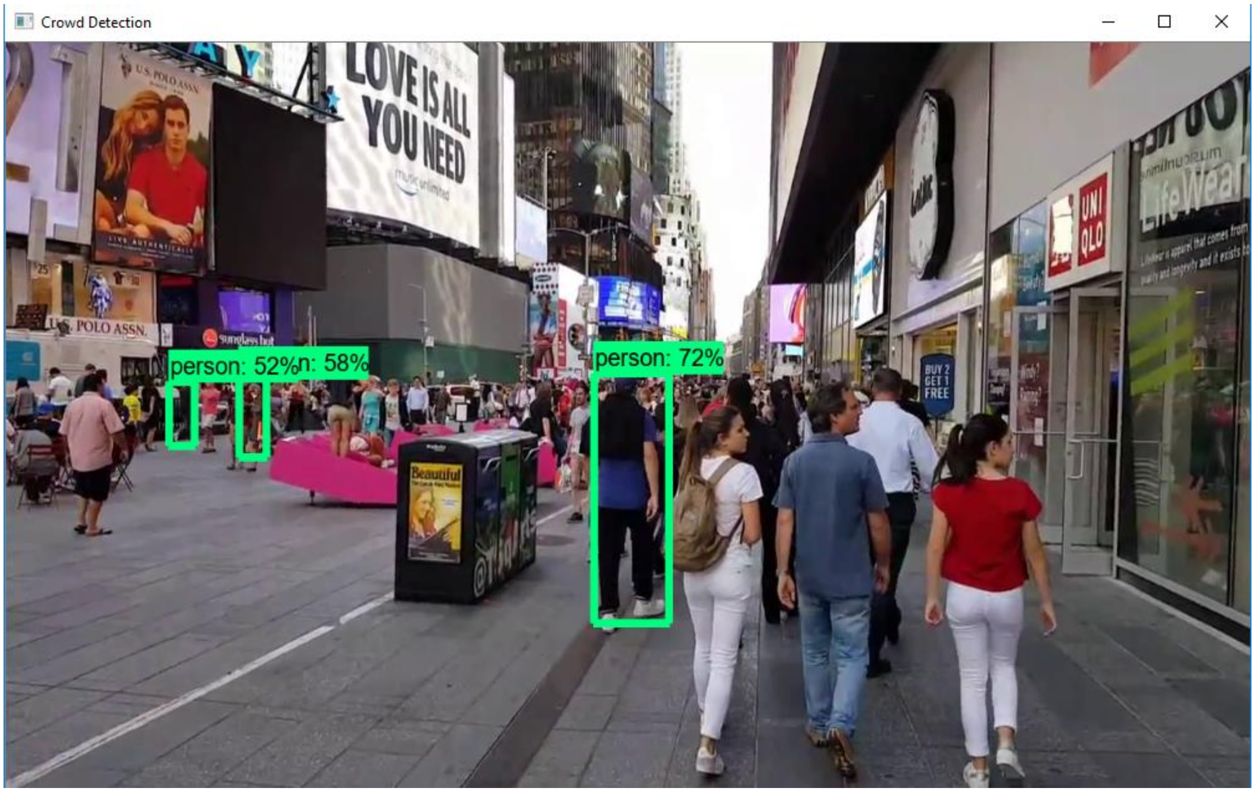
Deployment:

We are executing the python file with the command prompt as it needs to load the library and unzip the modules.

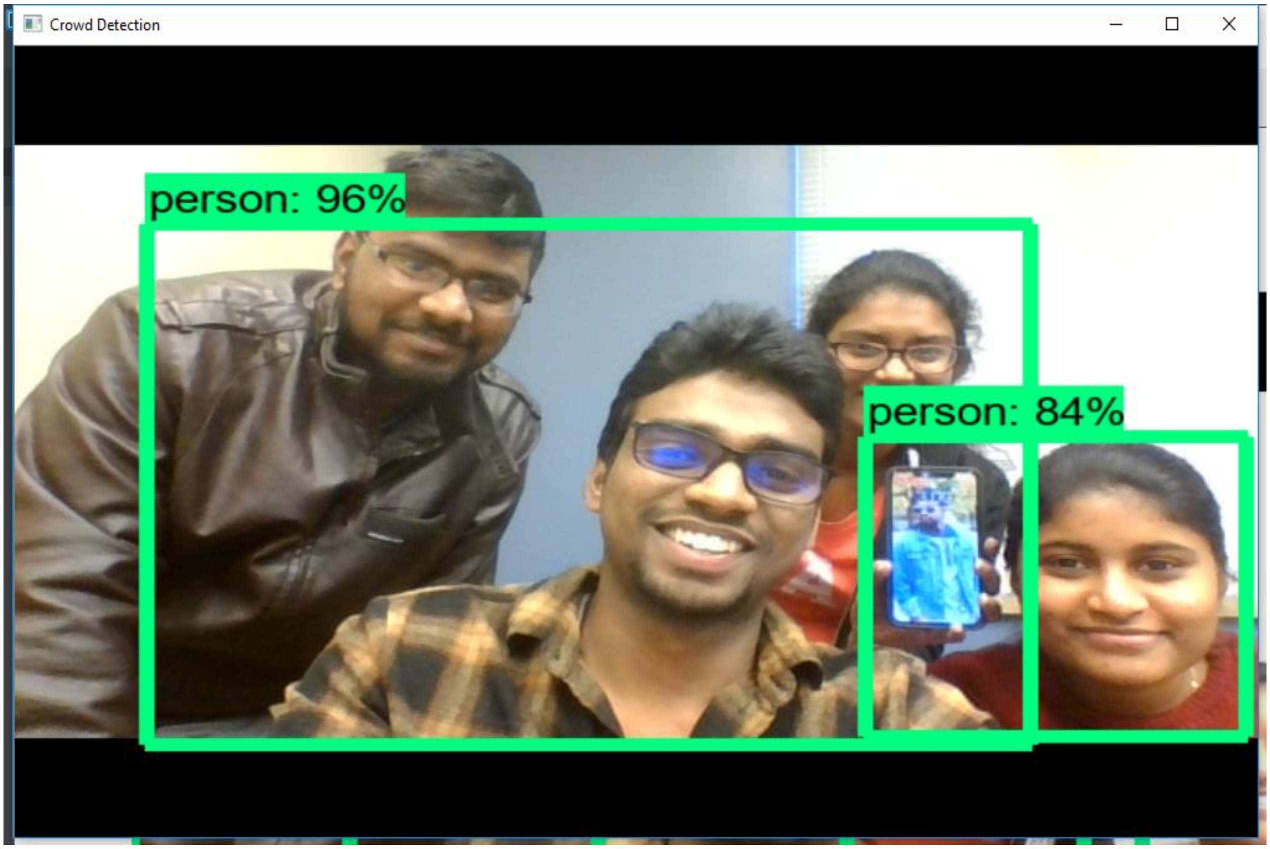

C:\Windows\System32\cmd.exe - python object_detection_crowd.py

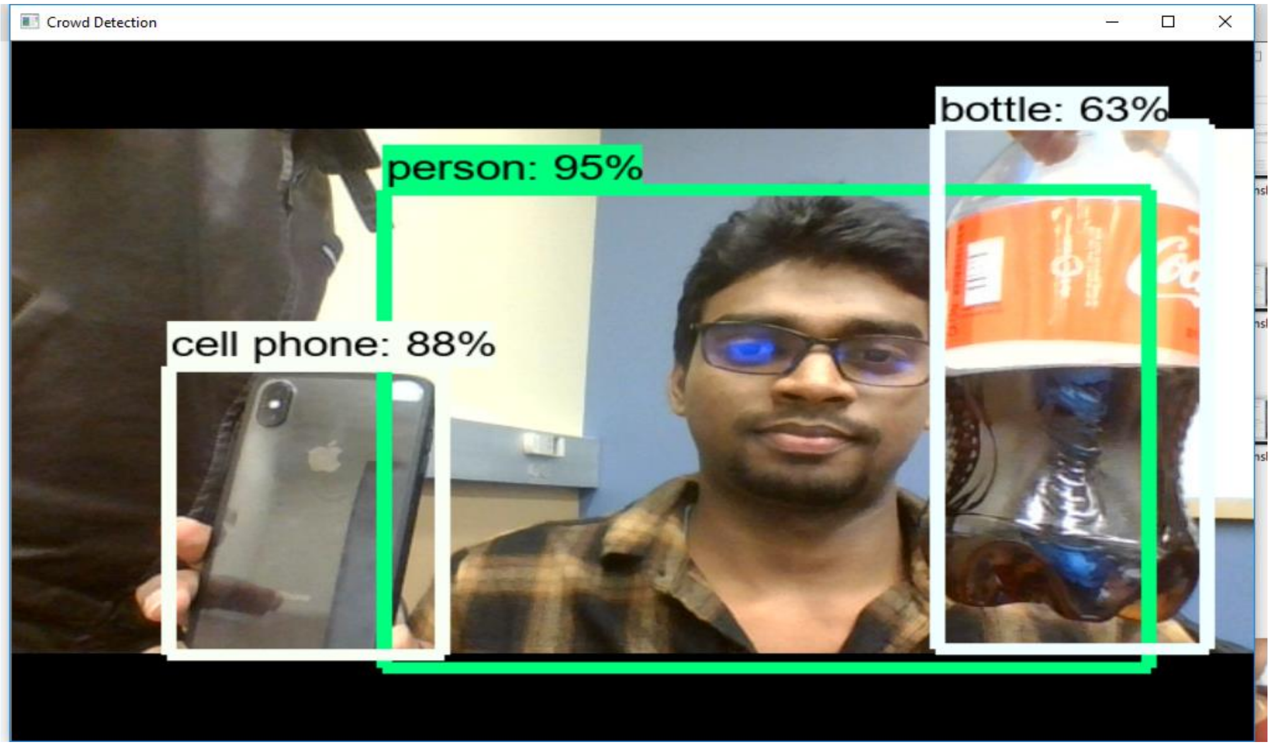
```
    run_metadata)
  File "C:\Users\sushm\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 1292, in _do_call
    return fn(*args)
  File "C:\Users\sushm\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 1277, in _run_fn
    options, feed_dict, fetch_list, target_list, run_metadata)
  File "C:\Users\sushm\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 1367, in _call_tf_sessionr
un
    run_metadata)
KeyboardInterrupt

C:\Users\sushm\Downloads>python object_detection_crowd.py
..\object_detection\utils\visualization_utils.py:27: UserWarning: matplotlib.pyplot as already been imported, this call
will have no effect.
  import matplotlib; matplotlib.use('Agg')  # pylint: disable=multiple-statements
2018-12-10 13:31:43.826220: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2

C:\Users\sushm\Downloads>python object_detection_crowd.py
..\object_detection\utils\visualization_utils.py:27: UserWarning: matplotlib.pyplot as already been imported, this call
will have no effect.
  import matplotlib; matplotlib.use('Agg')  # pylint: disable=multiple-statements
2018-12-10 15:04:24.905310: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2

C:\Users\sushm\Downloads>python object_detection_crowd.py
```

1.The output of the video.mp4

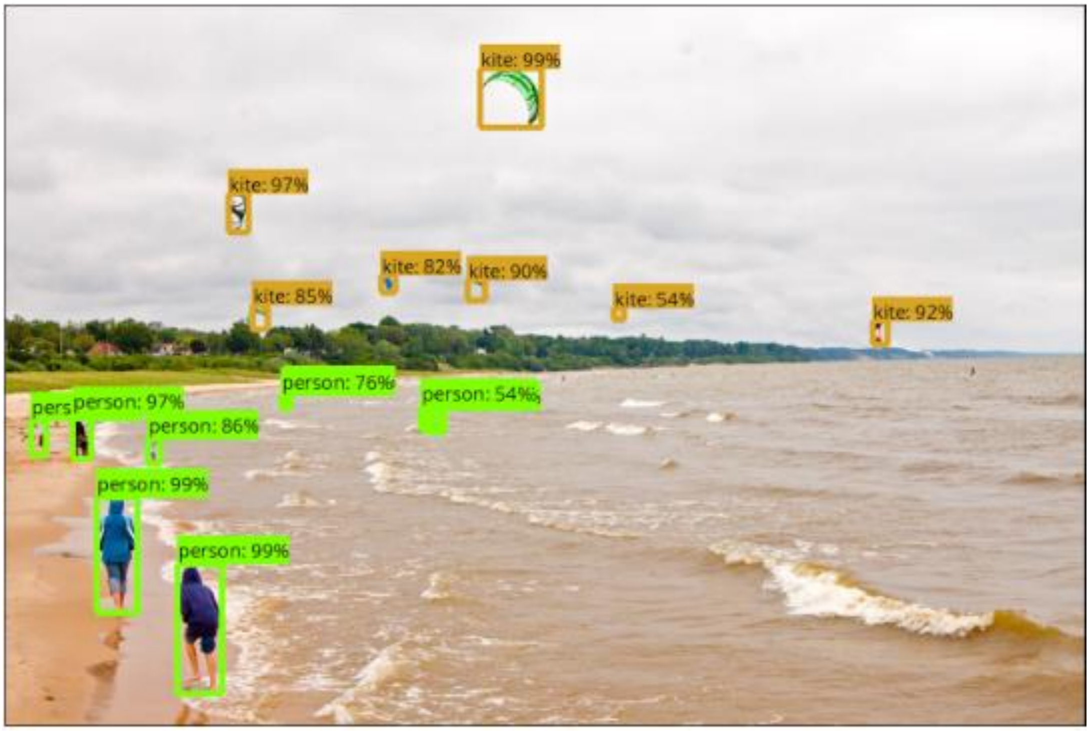


2.The output of the webcam





3.The output of the image.



Limitations:

- It does not detect all the objects in the screen, like the accuracy won't be 100%.
- The model should be trained, and the objects should be distinguished as it is an ongoing research the model will be improved with time.
- It also has some difficulty distinguishes between the same objects.

References:

https://github.com/tensorflow/models/tree/master/research/object_detection

https://medium.com/@WuStangDan/step-by-step-tensorflow-object-detection-api-tutorial-part-1-selecting-a-model-a02b6aabe39e

https://pythonprogramming.net/introduction-use-tensorflow-object-detection-api-tutorial/