**TEAM: PolyGon**

**TOOLS USED: OpenCV and Python**

#import the necessary packages

import numpy As np

import argparse

import cv2

#construct the argument parse And pase the arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--image", required=True,

help="path to input image")

ap.add\_argument("-p", "--prototxt", required=True,

help="path to Caffe 'deploy' prototxt file")

ap.add\_argument("-m", "--model", required=True,

help="path to Caffe pre-trained model")

ap.add\_argument("-c", "--confidence", type=float, default=0.2,

help="minimum probability to filter weak detections"

args = vars(ap.parse\_args())

# initialize the list Of Class labels MobileNet SSD was trained To

#detect, Then generate a Set Of bounding box colors For Each Class

CLASSES = ("background", "aeroplane", "bicycle", "bird", "boat",

"bottle", "bus", "car", "chair", "cow", "diningtable",

"dog", "horse", "motorbike", "person", "pottedplant", "sheep",

"sofa", "train", "tvmonitor")

COLORS = np.random,uniform(0, 225, size=(len(CLASSES), 3))

# load the serialized model from disk

print("(INFO) loading model...")

net = cv2.dnn.readNetFromCaffe(args("prototxt"), args("model"))

#load the input image And construct an input blob For the image

#by resizing To a fixed 300\*300 pixels And Then normalizing it

#(note: normalization Is done via the authors Of the MobileNet SSD

#implementation)

image = cv2.imread(args["image"])

(h, w) = image.shape[:2]

bolb = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300,300), 127.5

#pass the blob through the network And obtain the detections And

#predictions

print("[INFO] computing object detections...")

net.setInput(blob)

detections = net.forward()

# Loop over the detections

For i in np.arange(0,detection.shape[2]):

# extract the confidence(i.e., probability)associated with the

# prediction

confidence = detections[0,0,i,2]

# filter out weak detections by assuring the 'confidence' is

# greater than the minimum confidence

If confidence > args["confidence"]:

# extracts theindex of the class label from the 'detections',

# then compute the (x,y)-coordinates of the bounding box for

# for the object

idx = int(detections[0,0,i,1])

box = detections[0,0,i,3:7] \* np.array([w,h,w,h])

(startX,startY,endX,endY) = box.astype("int")

# display the prediction

label = "{}: {:.2f}%".format(CLASSES[idx], confidence \* 100)

print("[INFO] {}".format(label))

cv2.rectangle(image,(startX,startY), (endX,endY), COLORS[idx],2)

y = startY - 15 if startY - 15 > 15 else startY \* 15

cv2.putText(image,label,(startX,y),

cv2.FONT\_HERSHEY\_SIMPLEX,0.5,COLORS[idx],2)

# show the frame image

cv2.imshow("Frame",frame)

key2 = cv2.waitkey(1) & 0xFF

#If the Then 'q' key was pressed, break from the loop

if key == ord("q"):

break

#update the FPS counter

Fbs.update()

#.stop.the.timer.and.display.FPS.information

Fbs.stop()

print(“[INFO] elapsed time: {:.2f}”.format(fps.elapaed()))

print(“[INFO] approx.. FBS: {:.2f}”.format(fps.fps ()))

#do a bit of cleanup

Cv2.destroyAllWindows()

Vs.stop()

#USAGE

#python real\_time\_object\_detection.py –prototxt MobileNetSSD\_eploy.prototxt.txt --