Project Documentation

Object Detection using Yolo Algorithm

The objective of the Project:

The main objective of the following project is to train our model in such a way that it can detect as many objects as they can.

We have used the COCO dataset. Where COCO refers to a large-scale object detection segmentation and a captioning dataset that can detect about 80 objects.

*The following project is done under the supervision of the **verzeo** for the assessment of **Major Project** at verzeo for **Machine learning (June batch 1)**

*Following project is done **by Team ML06B1 – 5**

Introduction & Uses:

Object detection is a technology associated with laptop vision and image process that deals with sleuthing instances of objects.

Object detection is the craft of detective work in which instances of a specific category, like animals, humans and lots of a lot of them in a picture or video are identified.

Object detection algorithms usually leverage machine learning or deep learning to provide significant results.

Once humans verify pictures or videos, we will acknowledge and find objects of interest at intervals a matter of moments.

Uses:

Face detection:

Popular applications embody face detection, and other people count.
Have you ever detected; how facebook detects your face once you transfer a photo?
Face Detection is often a straightforward application of object detection that we tend to see in our lifestyle.

People count:

Object detection is often conjointly used for folks count.
Object detection is used for analyzing store performance or crowd statistics throughout festivals.
These tend to be more robust as folks move out of the frame quickly (also as a result of folks area unit non-rigid objects).

Vehicle detection:

Similarly, once the thing may be a vehicle like a bicycle or automotive, object detection with trailing will prove useful in estimating the speed of the item.
The type of ship getting into a port is often determined by object detection(depending on form, size, etc.).
This technique is presently in development in some European countries.

Manufacturing trade:

Object detection is additionally employed in industrial processes to spot products. Say you wish your machine to notice circular objects solely. Hough circle detection remodels often used for detection.

Brief Description:

The project has been planned meticulously by the team, and the planning was demarcated into the following phases.

1. Diving into Brief History

The case of object detection has broadly four main categories.

a) Semantic Segmentation
Semantic Segmentation deals with classifying each pixel belonging to a specific class.

b) Localization and Classification
Localization and Classification deal with, given an object to identify can we also put a border around it to identify the object.

c) Object Detection
Given a general setting, identify as many objects as possible and put a border around it.

d) Instance Segmentation
Create masks for individual objects in the image which is different from Semantic Segmentation.

The task is to apply techniques for Object Detection

2. Algorithms for Object Detection

A simple subset of Object Detection is the Localization and Classification(LAC). LAC has the essence of a regression problem which is solved by using CoVNets.

This idea of regression is mapped to Object Detection in the sense that we could naively run our algorithms over the image and search for the objects. Such type of approach has been further classified into known algorithms like FCNN, RCNN, FasterRCNN, Selective Search for Object Recognition, etc.

One such algorithm is YOLO(You look only once). Now since we have an idea of regression, many researchers have put in efforts to create a dataset of objects which are a common occurrence in the real world and hence came into existence the COCO Dataset.

3. Framework for the project

The project was decided to be built solely using Python and OpenCV library. The library was chosen because of its flexibility and the support for the task at hand.

Library Used:

OpenCV :

OpenCV is an open-source library that deals in computer vision and machine learning.
It comprises of more than 2500 optimized algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements. It has more than 47 thousand people's user community. The library is employed extensively in firms, research groups, and governmental bodies.
It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, and Macintosh Oos

YOLO :

Yolo is one of the best algorithms for Object Detection.

Its unified architecture is very fast.

It is based on regression, instead of selecting interesting parts of an image. It will predict the bounding boxes and class probabilities directly from the full image in one run of the algorithm.

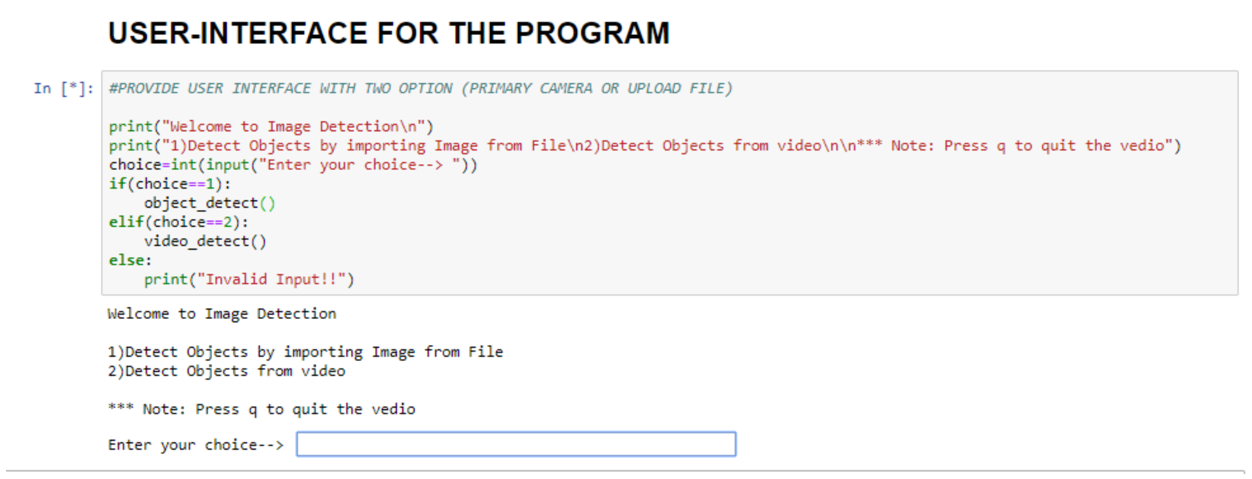
Since the full detection pipeline is a single network, it can be optimized end to end directly on detection performance.

Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes the image at 155 frames per second while still achieving double the mAP of other real-time detectors.

Compared to state-of-the-art detection systems, It makes more localization errors but is far less likely to predict false detections where nothing exists. It performs all other detection methods, including DPM and R-CNN, by a wide margin when generalizing from natural images to artwork on both the Picasso Dataset and the People-Art Dataset.

Design Document:

1) User Interface:



USER-INTERFACE FOR THE PROGRAM

```
In [*]: #PROVIDE USER INTERFACE WITH TWO OPTION (PRIMARY CAMERA OR UPLOAD FILE)

        print("Welcome to Image Detection\n")
        print("1)Detect Objects by importing Image from File\n2)Detect Objects from video\n\n*** Note: Press q to quit the vedio")
        choice=int(input("Enter your choice--> "))
        if(choice==1):
            object_detect()
        elif(choice==2):
            video_detect()
        else:
            print("Invalid Input!!")
```

```
Welcome to Image Detection

1)Detect Objects by importing Image from File
2)Detect Objects from video

*** Note: Press q to quit the vedio

Enter your choice-->
```

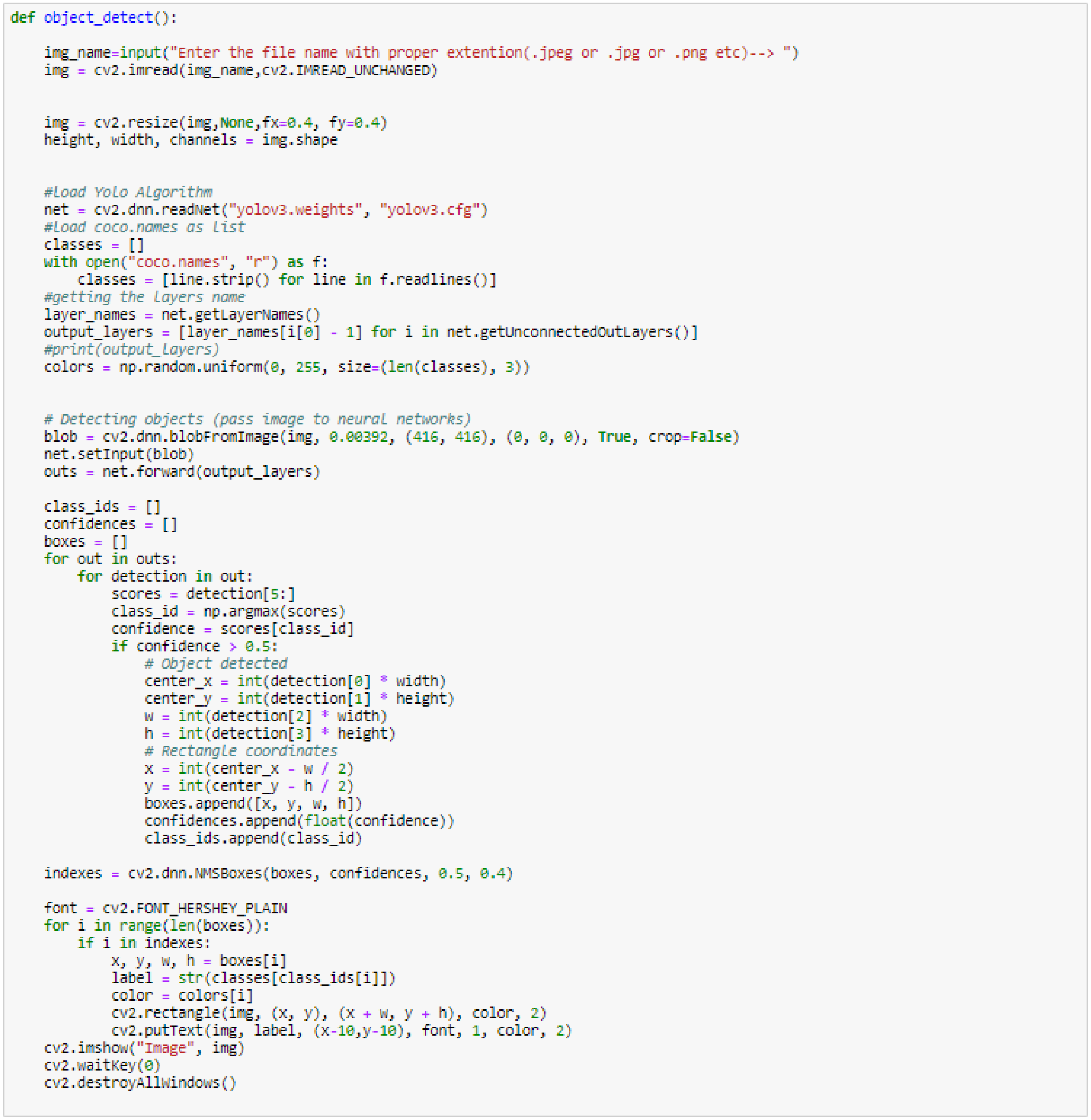
The user is been provided two options i.e

1) To detect an object by importing an image from file
2) Detect object from the video

The user gets the opportunity to choose among the following option
If they choose the first option then the user is asked to enter the name of the file with proper file extension (the separate function is called which detects the objects from the image).
if they choose the second option then the primary camera of the system is accessed and continues detection of the object takes place (the separate function is called that detects the objects from the video).

For the first condition, there is no termination condition for the second condition on pressing q from the keyboard will quit the video.

2) **Detect Object From the Image**:



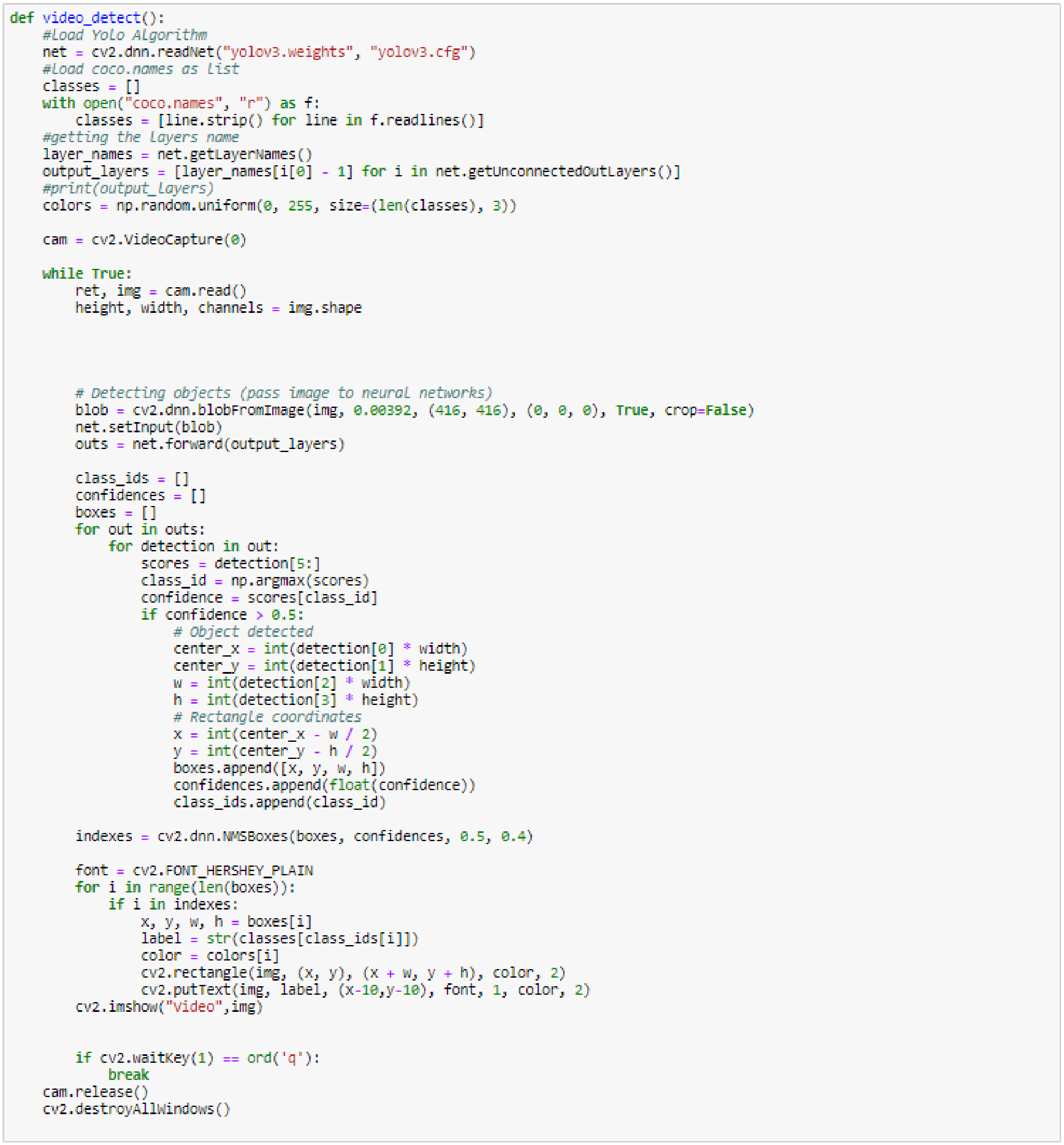
```python
def object_detect():

    img_name=input("Enter the file name with proper extention(.jpeg or .jpg or .png etc)--> ")
    img = cv2.imread(img_name,cv2.IMREAD_UNCHANGED)


    img = cv2.resize(img,None,fx=0.4, fy=0.4)
    height, width, channels = img.shape


    #Load Yolo Algorithm
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    #Load coco.names as list
    classes = []
    with open("coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
    #getting the layers name
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
    #print(output_layers)
    colors = np.random.uniform(0, 255, size=(len(classes), 3))


    # Detecting objects (pass image to neural networks)
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    font = cv2.FONT_HERSHEY_PLAIN
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            color = colors[i]
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
            cv2.putText(img, label, (x-10,y-10), font, 1, color, 2)
    cv2.imshow("Image", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

The following function first asks for the user to input the file name with the proper extension in case the extension doesn't match or the file name doesn't match or file doesn't exist in the same directory it throws an error.

The following function after reading the image it loads the algorithm (Yolo). Later the detection of the objects takes place using the deep neural network (DNN).After getting the number of objects using the OpenCV we place a rectangular container around the object since the objects are detected from coco.names file we get the name of the object and place above it.

After completing this process the image shown with the proper detection of the object.

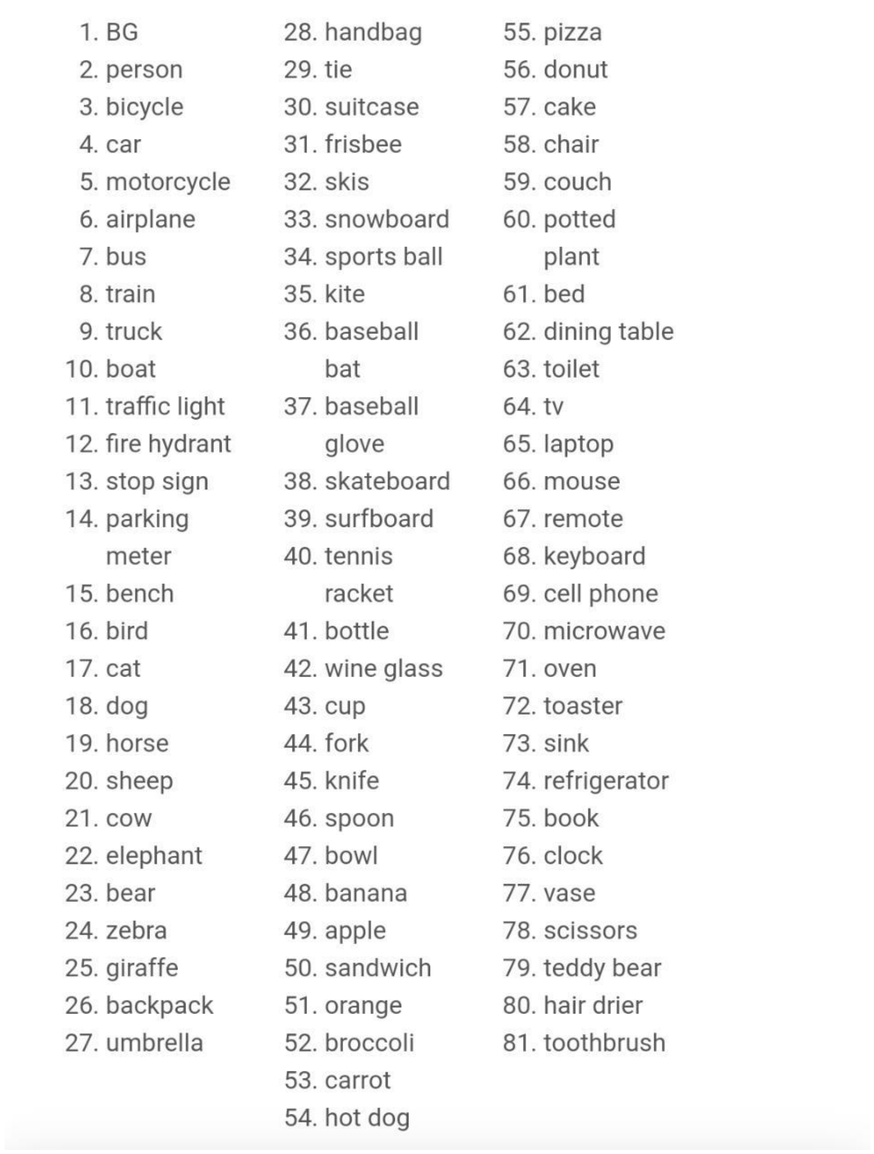
3) **Detection of an object using video**:



```
def video_detect():
    #Load Yolo Algorithm
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    #Load coco.names as list
    classes = []
    with open("coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
    #getting the layers name
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
    #print(output_layers)
    colors = np.random.uniform(0, 255, size=(len(classes), 3))

    cam = cv2.VideoCapture(0)

    while True:
        ret, img = cam.read()
        height, width, channels = img.shape




        # Detecting objects (pass image to neural networks)
        blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
        net.setInput(blob)
        outs = net.forward(output_layers)

        class_ids = []
        confidences = []
        boxes = []
        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > 0.5:
                    # Object detected
                    center_x = int(detection[0] * width)
                    center_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)
                    # Rectangle coordinates
                    x = int(center_x - w / 2)
                    y = int(center_y - h / 2)
                    boxes.append([x, y, w, h])
                    confidences.append(float(confidence))
                    class_ids.append(class_id)

        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

        font = cv2.FONT_HERSHEY_PLAIN
        for i in range(len(boxes)):
            if i in indexes:
                x, y, w, h = boxes[i]
                label = str(classes[class_ids[i]])
                color = colors[i]
                cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                cv2.putText(img, label, (x-10,y-10), font, 1, color, 2)
        cv2.imshow("Video",img)


        if cv2.waitKey(1) == ord('q'):
            break
    cam.release()
    cv2.destroyAllWindows()
```

The following function does a similar activity as the previous function only difference is that the camera is running until the exit condition is met which is the pressing of keyboard key “q”. The image of each frame is continuously obtained. The algorithm will be trained at the beginning itself. The object detection and placing rectangular containers around the object using OpenCV and the object names are obtained by the coco.names file.

The following model detects 81 objects they are:



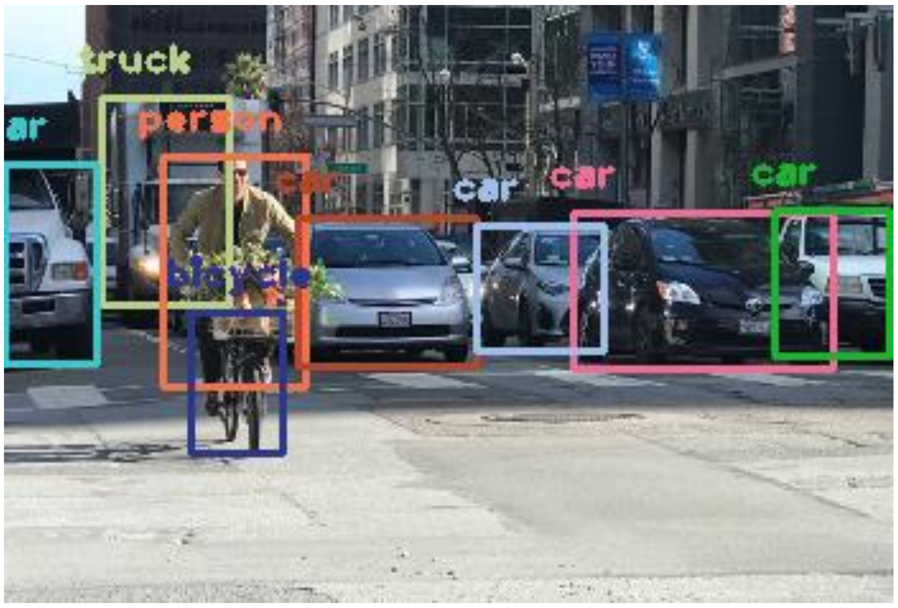
1. BG
2. person
3. bicycle
4. car
5. motorcycle
6. airplane
7. bus
8. train
9. truck
10. boat
11. traffic light
12. fire hydrant
13. stop sign
14. parking meter
15. bench
16. bird
17. cat
18. dog
19. horse
20. sheep
21. cow
22. elephant
23. bear
24. zebra
25. giraffe
26. backpack
27. umbrella
28. handbag
29. tie
30. suitcase
31. frisbee
32. skis
33. snowboard
34. sports ball
35. kite
36. baseball bat
37. baseball glove
38. skateboard
39. surfboard
40. tennis racket
41. bottle
42. wine glass
43. cup
44. fork
45. knife
46. spoon
47. bowl
48. banana
49. apple
50. sandwich
51. orange
52. broccoli
53. carrot
54. hot dog
55. pizza
56. donut
57. cake
58. chair
59. couch
60. potted plant
61. bed
62. dining table
63. toilet
64. tv
65. laptop
66. mouse
67. remote
68. keyboard
69. cell phone
70. microwave
71. oven
72. toaster
73. sink
74. refrigerator
75. book
76. clock
77. vase
78. scissors
79. teddy bear
80. hair drier
81. toothbrush

The project Outcome:

1) Detect Object using image:

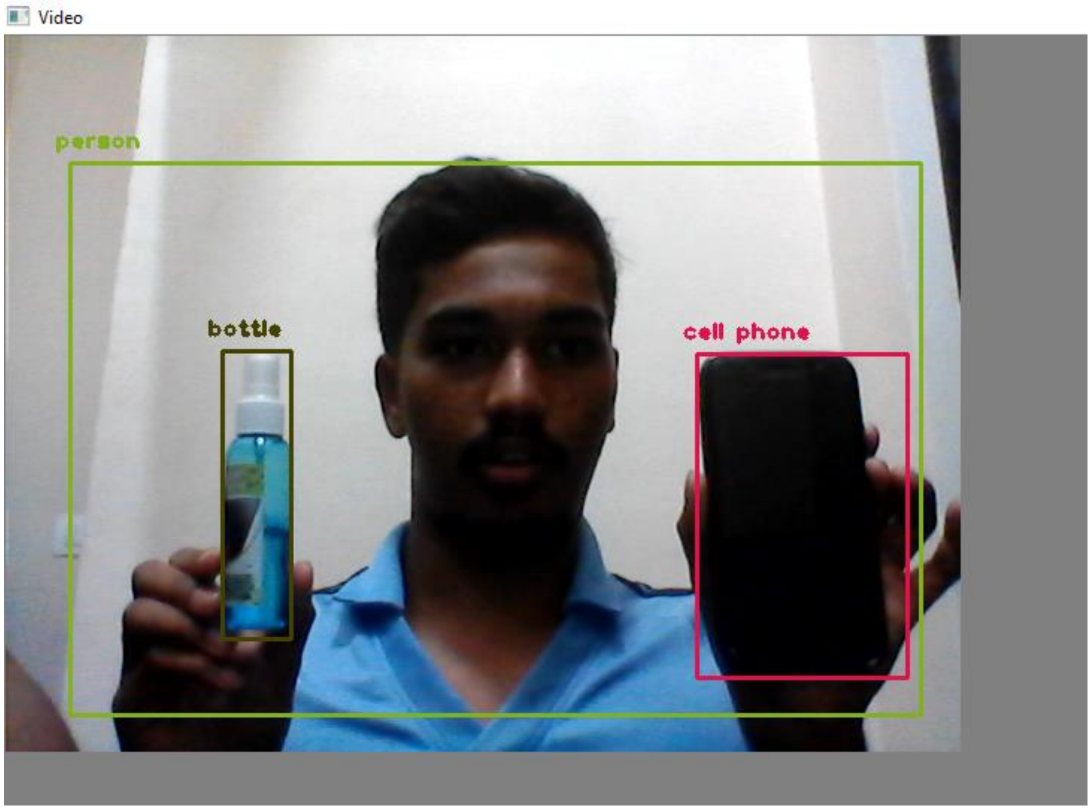
i) Input File:

ii) Output:



2) Detect an object using video:

Input/Output : (remains in the same video)



Conclusion:

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision and deep learning. The custom dataset was created using labeling and the evaluation was consistent. This can be used in real -time applications that require object detection for pre-processing in their pipeline.
An important scope would be to train the system on a video sequence for usage in tracking applications. The addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.