Detection Of Real-Time Objects

Abstract

The process of finding real-world objects using different computational techniques is known as object detection. It helps us in understating the object's images in a better manner. Recently, the need for multiple object detection has risen in many applications because of its many applications like security, surveillance, automatic driver cars .etc. Nevertheless, the performance of the already existing techniques used to apply to the real-world image data was very low and bad for imagerelated tasks like object detection and recognition, etc. So, in this project, we will apply a computer vision and advanced image processing technique known as OpenCV to increase the performance of image-related data and tasks [1]. We will propose a solution using the evolutional image processing technique which can improve the performance for these tasks and can also produce the same performance if the data is restricted. We will access and produce our model according to the recent standards for object detections and predictions and will show how well it will perform as compared

Chapter 1 Introduction

1.1. Introduction

The purpose of object detection is to identify and provide us the vital information which helps us in understanding certain images or videos of the objects. In recent years because of the advancement in the internet and technologies, multiple object detection act as an important need of the day in a variety of applications. In the past decades, there is no such gain in the ability to detect or predict multiple objects using image data in our applications accurately [2]. This task of object recognition is still a mystery. In these past years, different methods are used to detect and predict multiple objects by using different image features, for example, some of these applications use classification algorithms made specifically for this kind of task.

Many Machine learning techniques are applied in the past but they cannot meet the challenge of different imaging situations and errors like low-resolution images, different types of angles and variations in the features of the objects, and many more. As the object image feature is a key for this purpose of prediction and detection of the objects. In recent years the boom of Artificial Intelligence, computer vision, and image processing techniques have shown great development in the field of object recognition and predictions. In this project, we intend to make a revaluation of automatic object detection in real-time.

In this project, we will produce a different but simple method that can be more reliable and accurate than the past methods. Our method will be applied and tested on the most recent object datasets and it will show that the complex imaging conditions and the simplest road map of our method will show the great gap of improvement in this field as compared to the past [3]. This technique will open the doors to more advancement and improvement in the field of multiple object detection fields.

1.2. Overall Description

The overall description of the project is as follows: -

1.2.1. Objectives

The objectives of this project are:

- To design a system able to simultaneously detect multiple objects on real-time data.
- These multiple objects include persons, bicycles, cars, motorcycles, airplanes, buses, train trucks, boats, traffic lights, fire hydrants, etc.

- To identify and locate multiple effective targets in real-time first it will train on the dataset which includes images of the different objects.

1.2.2. Problem Description

The problem description of this project is as follows:

- In the past, many techniques are applied to the problem of multiple object detection but they are unable to perform well because of the complexity of the different image features, for example, low-resolution images, different kinds of variations in features, and many more. As the image features are key for this purpose of detection of multiple objects.
- Recently, the use of intelligent security applications is increasing due to the rapid increase in crimes and so there is a need for a framework that can meet today's object detection needs.
- It is a need of the day to build such a system which works on the methodology of latest AIbased algorithms and a Computer vison algorithm to detect the multiple objects quickly in no time with accuracy ratio.

1.2.3. Methodology

We use the images dataset of different objects in this project which has 91 object categories of different real-time world things like cars, traffic signals, dogs, cats, etc. In this project, we propose deep learning embedded with computer vision architecture that uses a Deep Convolutional Neural Network (CNN) for the complex imaging features extraction because in the past the main reason for the failure of the methods used for the objects detection problems is the complexity of the different images features [4].

The CNN will perform outstandingly for this purpose and it is more reliable for feature extraction from complex images than the other method used in the past. Our deep learning model allows us to focus on the complex and vital informative parts of the images. we will then use the computer vision technique for real-time object detection. OpenCV works best for real-time data and gives the best performance for this purpose.

1.2.4. Product Scope

The need for automatic multiple object detection is the need of the day in many security and other applications of today’s world because of its many applications in lots of fields for example security surveillance, video conferencing .etc. So, we will create such a framework that aids all the applications needed to detect multiple objects from images in real-time. In this project, we intend to make a revaluation in automatic detection of multiple objects and we will make simple but effective architecture and framework to efficiently detect multiple objects from images dataset.

1.2.5. Working Atmosphere

The system will need the following software and hardware requirements.

 Minutest System Wants ➢ **Operating Systems requirements:** Windows 7 or above

➢ **CPU:** Intel Core i5 3rd generation processor or better

1.3. Expectations and requirements

Our system needs a good pre-tained dataset to run the system environment. An AI technology will be used to automate the environment to detect the object in realtime will the

1.4. External System Needs

The external system needs are as follows:

1.4.1. Hardware

 Any Computer With minutest requirements

1.4.2. Software

The problem of extracting multiple products from photos is based on AI technology known as deep learning and computer vision. The algorithm will be created using the following tool:

- Anaconda 3 2020.11(64 bit)
- Jupiter notebook
- PyCharm

1.5. Development Functions

A DL-based approach with a Computer Vision technique OpenCV used for the detection of Multiple objects in real-time. The taught features are more discriminative and concise than the features that were manually constructed since the DL algorithm can learn the features in a hierarchical manner, and OpenCV will recognise many objects in real-time.

1.5.1. Description and Priority

It will train on the dataset of different objects images then extract features from it using deep learning models then it will use OpenCV for the detection of multiple objects in real-time.

1.5.2. Response Sequences

When the real-time data is shown from some device, the system will detect the objects in that data.

Chapter 2 Literature Review

2.1. Introduction

The research on the topic of object detection and estimation has been going on for the past few years. Many solutions have been introduced during the time with a significantly low level of gain and low results. In the early ages, the research focused on manually classifying and extracting features from image details and using the difference in these features for prediction purposes.

In the past, the research on object detection and estimation can be found the early 1990s when neural networks are used for this purpose [2]. After that, we found that in the early 2000s deep learning algorithms were used for multiple object detection purposes and they gained some accuracy when predicting the blur images. However, the bad implementation and characterization of their data stop their systems from getting improvement in performance and accuracy for the complex imaging data. But every method was able to serve a single object at a time.

So, for the first time in history, this custom was changed in early 2015 [2], when a framework was generated by researchers which can detect multiple objects detection and they also found that every real-world image has different features and different alignment settings. They focus on simplifying the image before getting it to classify like its different features and other things and they use a different variety of angles when they trained their images for this purpose they use different computer vision techniques. These techniques help them gain significantly good performance as compared to other methods.

2.2. Related Works

Machine learning is a technique in which we can manually feature extraction tasks and used it for many purposes like classification, prediction problems, etc. Whereas deep learning can automatically extract features from the dataset and best for many images related problems. In the past, the research on object detection and estimation can be found the early 1990s when neural networks are used for this purpose [2]. After that, we found that in the early 2000s deep learning algorithms were used for multiple object detection purposes and they gained some accuracy when predicting the blur images. However, the bad implementation and characterization of their data stop their systems from getting improvement in performance and accuracy for the complex imaging data. But every method was able to serve a single object at a time.

In the past since 2012, researchers have found that these deep learning algorithms perform well and give good performance than the other algorithm for image-related problems. A researcher named A. Krizhevsky et al. [4] were able to find a method that uses a CNN for the classification of the images having a range of up to 1000 classes achieving a test error rate below 15.3% which was a good improvement. Many early techniques are applied in the past but they cannot meet the challenge of different imaging situations and errors like low-resolution images, different kinds of variations in the image features, and many more.

As the image features are the key for this purpose of prediction and detection. For example in early 1999, [4]Kwon and Lobo fucoses on the different variations in the features of images like dimensions and were able to find a fast solution for estimating object problems. But there were some flaws in their system. So, in 2004, a researcher name Lanita et al. give the solution to this problem by focusing on both geometrical and texture features by using a technique known as the

Active Appearance Model (AMM). This technique does not produce good results if the dataset includes different variational object images.

2.2.1. Terminology

The terms we are using in this system are mentioned below:

- **Computer Vision:** OpenCV
- **Deep Learning Models:** Convolutional Neural Network, Single Shot Detector (SSD)
- **Domain:** Deep Learning & Computer Vision
- **Dataset**: Multiple Objects Dataset
- Multiple Objects

2.2.2. Categorization of Existing Techniques/Works/Research

The technologies or techniques used in past are single purposed methodologies are past AI-based algorithms used for classification and detection of the multiple objects, now we are using different and latest AI-based techniques which are more reliable and more accurate.

2.2.3. Proposed Improvements in Existing Works

We have entered a new era of AI and image processing and our system takes a lot of benefits from this new era and using the latest algo of image processing and AI to solve this problem of multiple object detection. This system speed of detecting the multiple objects is significantly fast and more accurate. This will save the time and make the user more satisfied than ever before.

2.3. Summary

We have found that both the models Single- shot Detector and an OpenCV are used to design such a system in which it will automatically detect the multiple objects in real-time and SSD is a pretrained model used for the detection purposes and it will detect above ninety classes. Our system has more accuracy and it is very reliable as compared to the past systems

Chapter 3 System Design

3.1. Diccussion

The project's system design for multiple objects detection will be covered in this chapter. The project is mostly a research project, hence there has very little user interface. As stated in the introductory chapter, the project's goal is to develop a Computer Vision and DL algorithm that automates the detection of many objects in real-time.

3.1.1. System Outline

The system is developed using Computer Vision and Deep Learning techniques to detect multiple objects without any human effort. The system uses pre-trained dataset to automatically detect the multiple objects using AI algorithms and the dataset contains different classes of objects.

3.1.2. Project Design Plan

The pre-trained images will be given to the system and then the algorithms will be applied to get the accurate result from the system and the system will detect the multiple objects through live camera or from real-time device.

3.2. Project Deliberations

This project has no design so there is no need for any design consideration required.

3.2.1. Expectations

The system will detect the multiple objects in real-time from camera and it uses pre-trained dataset and multiple algos for that.

3.2.2. Restrictions

System required a image data to train on and accurately detect the objects. The other form of data will not be applicable in this system.

3.3. Activity Diagram

The below figure 3.1 shows the operational process of this project and the steps of how it will work and detect multiple objects in real-time.

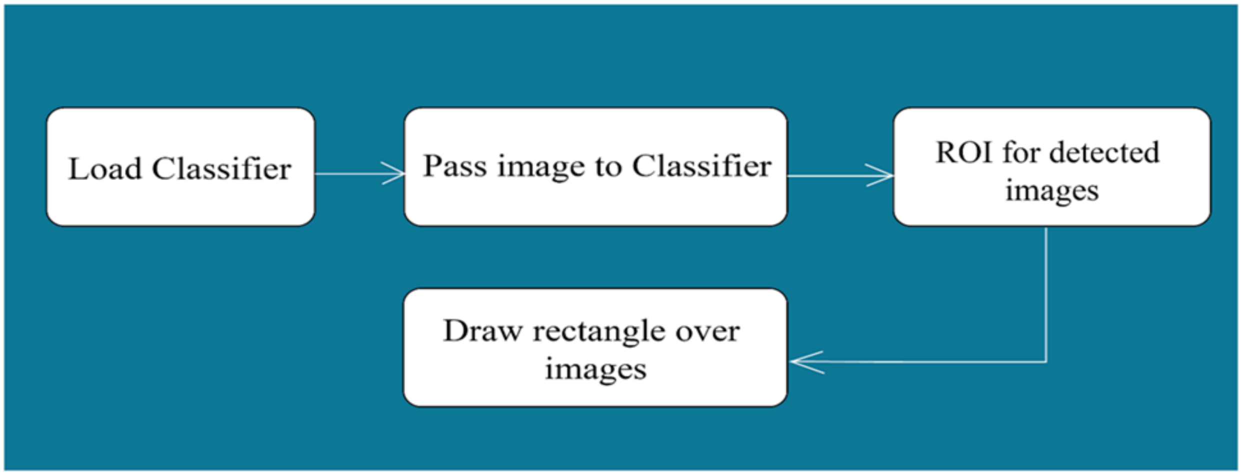


Figure 3. 1 Workflow Diagram

Figure 3.1 is a Workflow diagram of the project which shows the different stages of the flow of the project. Below shown in figure 2 is the activity diagram which shows the working of our project in very simple steps.

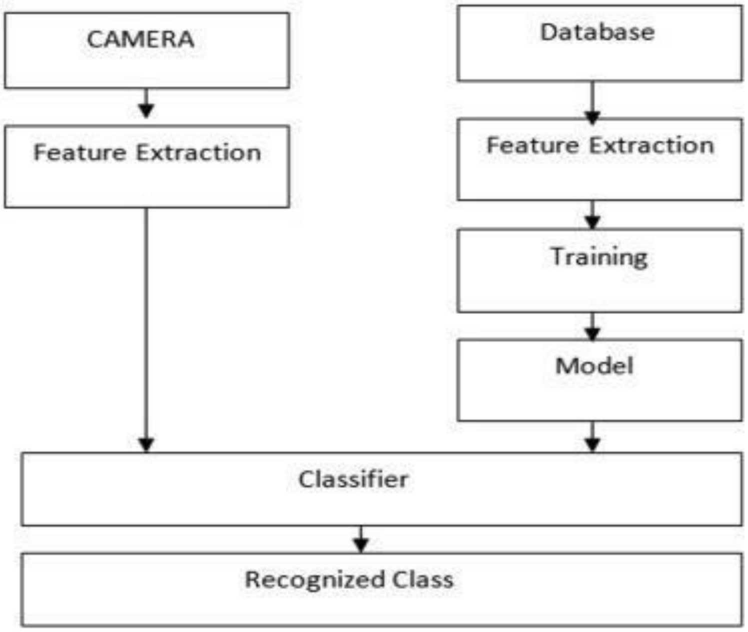


Figure 3. 2 Activity Diagram

Chapter 4 Implementation

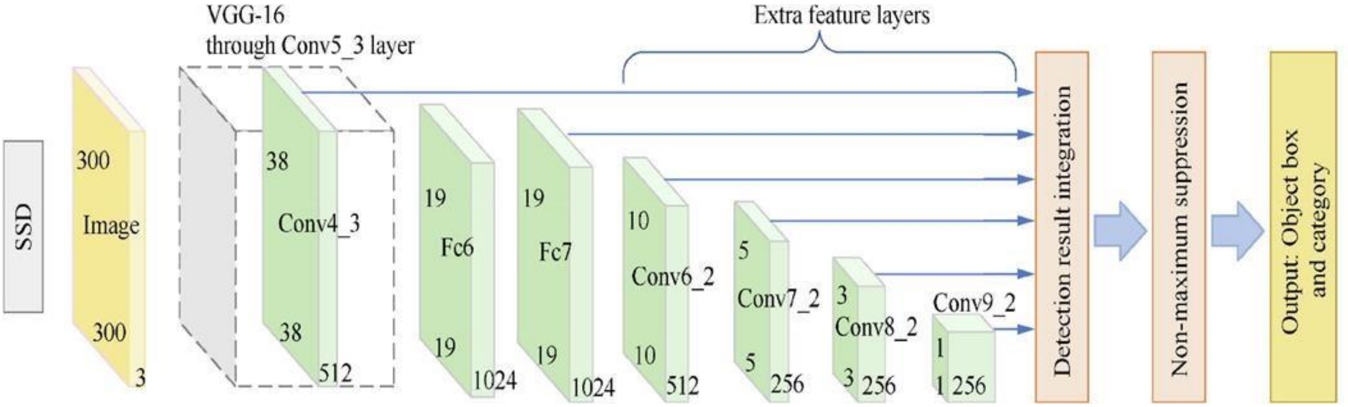
4.1. Arguments

Different images of multiple objects dataset which has different images of different objects.

Because this is a classification problem and datasets are based on images, we use a Single Shot Detector (SSD) for the detection of multiple objects. On the first phase, our model can detect 92 classes of different objects upon entering the desired object image into the model for detection purposes.

4.2. Development Methodologies

This system is for the detection of multiple objects and in this part of the documentation we will diccuss the steps of development of this system. In the first phase, we enter the desired object into the model, and after that the DL based algo SSD will be used to detection of multiple objects.



Below is the working architecture of the project and it is explained in the next section.

Figure 4. 1 Development Methodology

4.3. Execution Apparatuses and Techs

The project is build using a programing language used for AI project known as Python. We have the information and knowledge for the implementation of this project. Also we have studied these algorithms in the university so we are ready to build this project.

4.3.1. Tech Tools

Below mentioned software, hardware and algos are used to build this project.

□ **Hardware**

➢ Any kind of Computer Machine equipped with the requiremnets

□ **AI Algos**

➢ CNN (Convolutional Neural Network)

➢ SSD (Single Short Detector)

➢ OpenCV

□ **Software**

➢ Python

➢ Anaconda as IDE.

4.3.2. Single Shot Detector

SSD is an object detection model that computes the bounding box and category of an object from an input image. This Single Shot Detector (SSD) object detection model uses Mobile-net as the backbone and can achieve fast object detection optimized for mobile devices. A single shot detector takes only one shot to detect multiple objects present in an image using a multi-box. It is significantly faster in speed and high-accuracy object detection algorithm.

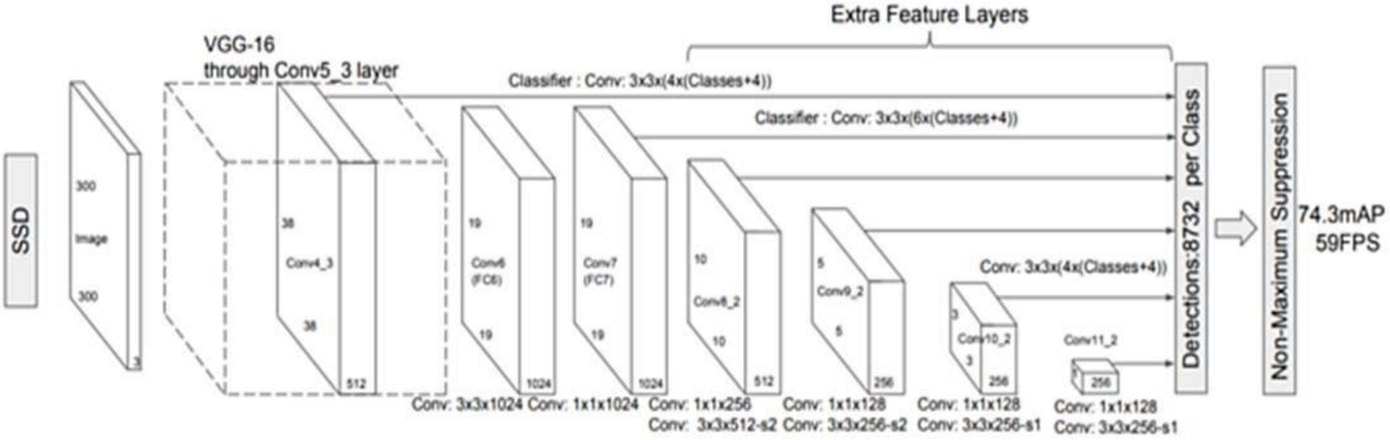


Figure 4. 2 SSD

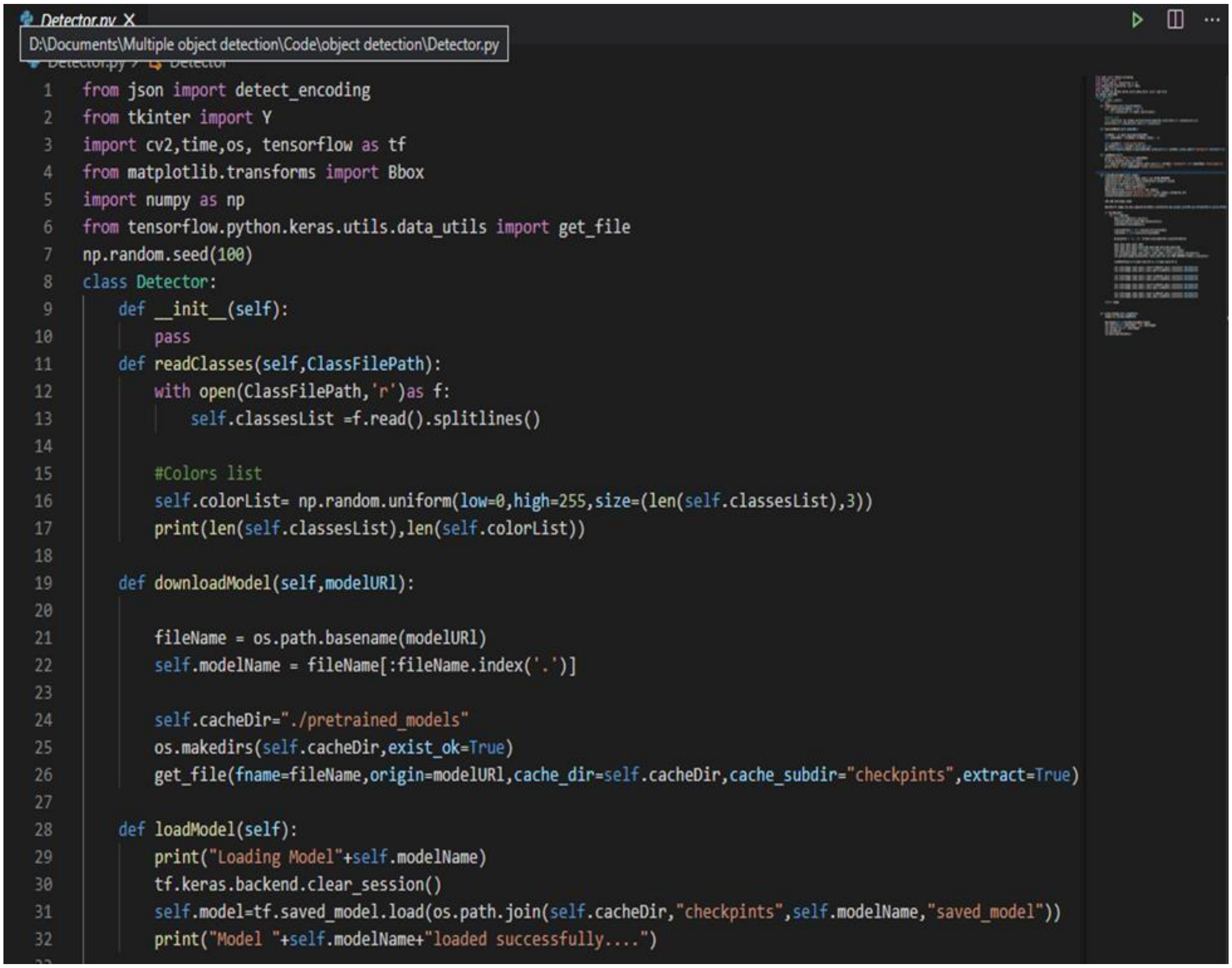
SSD takes a (3,300,300) image as input and outputs (1,3000,4) boxes and (1,3000,21) Boxes contain offset values (cx, cy, w, h) from the default box. Scores contain confidence values for the presence of each of the 20 object categories, the value 0 is reserved for the background. In SSD, after extracting the features using an arbitrary backbone, the bounding boxes are calculated at each resolution while reducing the resolution with Extra Feature Layers. SSD will concatenate the output of the six levels of resolution and calculate a total of 3000 bounding boxes, and finally, filter out bounding boxes using non-maximum suppression (NMS).

4.4. Implementation

The building process of the system is diccuss below:

4.5. Dataset

We use an image dataset that has different object images. Because this is a classification problem and dataset are based on images, we use Single Shot dec for the detection of multiple objects. The dataset includes different images of 92 different classes of different categories. At the start of the implementation, we use the pre-trained model for the dataset which is used for the dataset. The figure below shows the code for the pre-trained model:



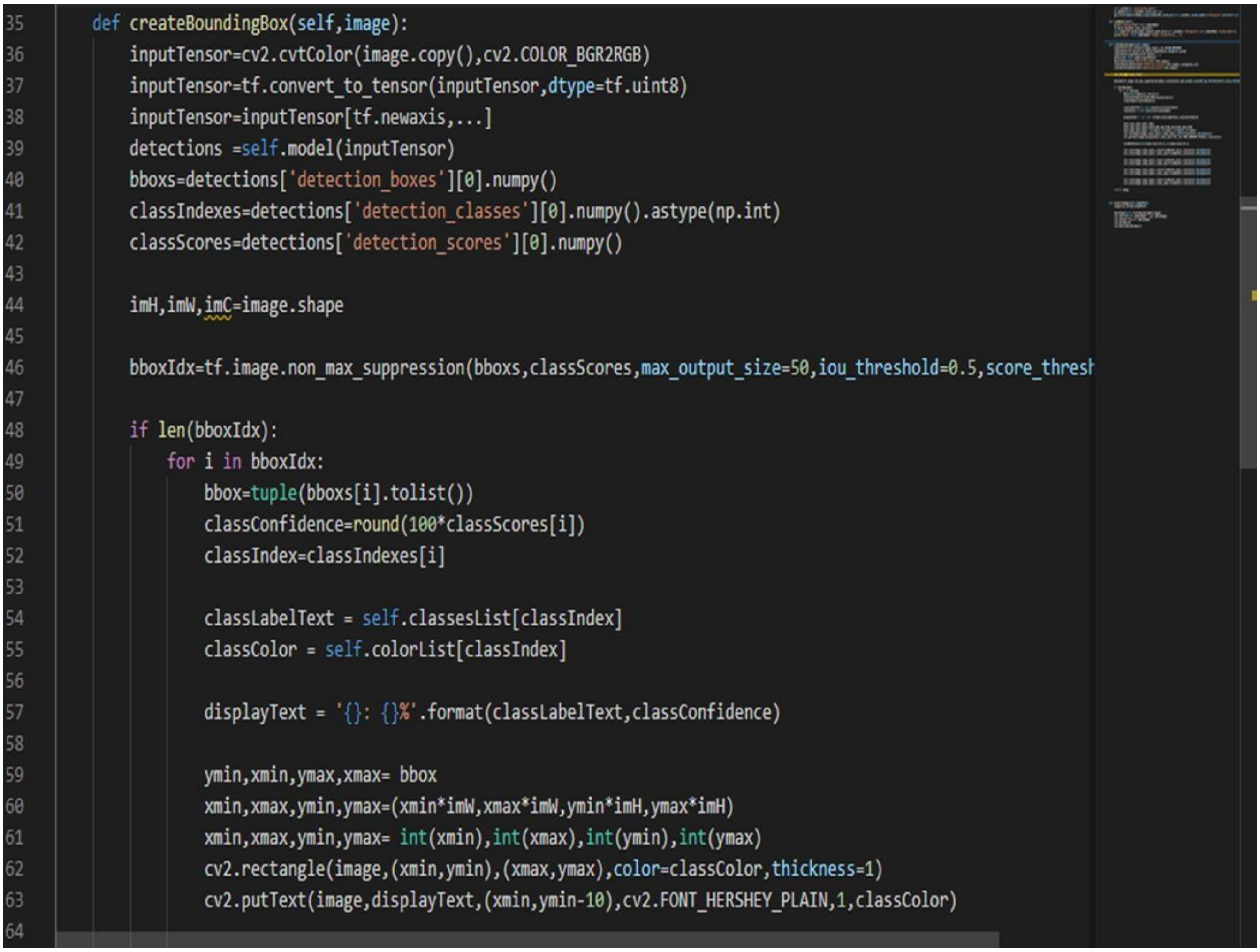
Detector.py X
D:\Documents\Multiple object detection\Code\object detection\Detector.py
Detector.py > Detector

```python
from json import detect_encoding
from tkinter import Y
import cv2,time,os, tensorflow as tf
from matplotlib.transforms import Bbox
import numpy as np
from tensorflow.python.keras.utils.data_utils import get_file
np.random.seed(100)
class Detector:
    def __init__(self):
        pass
    def readClasses(self,ClassFilePath):
        with open(ClassFilePath,'r')as f:
            self.classesList =f.read().splitlines()

        #Colors list
        self.colorList= np.random.uniform(low=0,high=255,size=(len(self.classesList),3))
        print(len(self.classesList),len(self.colorList))

    def downloadModel(self,modelURl):

        fileName = os.path.basename(modelURl)
        self.modelName = fileName[:fileName.index('.')]

        self.cacheDir="./pretrained_models"
        os.makedirs(self.cacheDir,exist_ok=True)
        get_file(fname=fileName,origin=modelURl,cache_dir=self.cacheDir,cache_subdir="checkpints",extract=True)

    def loadModel(self):
        print("Loading Model"+self.modelName)
        tf.keras.backend.clear_session()
        self.model=tf.saved_model.load(os.path.join(self.cacheDir,"checkpints",self.modelName,"saved_model"))
        print("Model "+self.modelName+"loaded successfully....")
```

Figure 4. 3 Pre-trained Model

4.5.1. Single Shot Detector

Now we apply SSD on the pre-trained model which will do the features extraction, then it make a bounding box around the object and detect the desired multiple objects:



```
    def createBoundingBox(self,image):
        inputTensor=cv2.cvtColor(image.copy(),cv2.COLOR_BGR2RGB)
        inputTensor=tf.convert_to_tensor(inputTensor,dtype=tf.uint8)
        inputTensor=inputTensor[tf.newaxis,...]
        detections =self.model(inputTensor)
        bboxs=detections['detection_boxes'][0].numpy()
        classIndexes=detections['detection_classes'][0].numpy().astype(np.int)
        classScores=detections['detection_scores'][0].numpy()

        imH,imW,imC=image.shape

        bboxIdx=tf.image.non_max_suppression(bboxs,classScores,max_output_size=50,iou_threshold=0.5,score_thresh

        if len(bboxIdx):
            for i in bboxIdx:
                bbox=tuple(bboxs[i].tolist())
                classConfidence=round(100*classScores[i])
                classIndex=classIndexes[i]

                classLabelText = self.classesList[classIndex]
                classColor = self.colorList[classIndex]

                displayText = '{}: {}%'.format(classLabelText,classConfidence)

                ymin,xmin,ymax,xmax= bbox
                xmin,xmax,ymin,ymax=(xmin*imW,xmax*imW,ymin*imH,ymax*imH)
                xmin,xmax,ymin,ymax= int(xmin),int(xmax),int(ymin),int(ymax)
                cv2.rectangle(image,(xmin,ymin),(xmax,ymax),color=classColor,thickness=1)
                cv2.putText(image,displayText,(xmin,ymin-10),cv2.FONT_HERSHEY_PLAIN,1,classColor)
```

Figure 4. 4 SSD

4.5.2. Result

In the end, the model will predict the object in the images entered by the user and also tells the confidence level of the result. Below we will show the different images we entered in the project and the output it produced are following.:

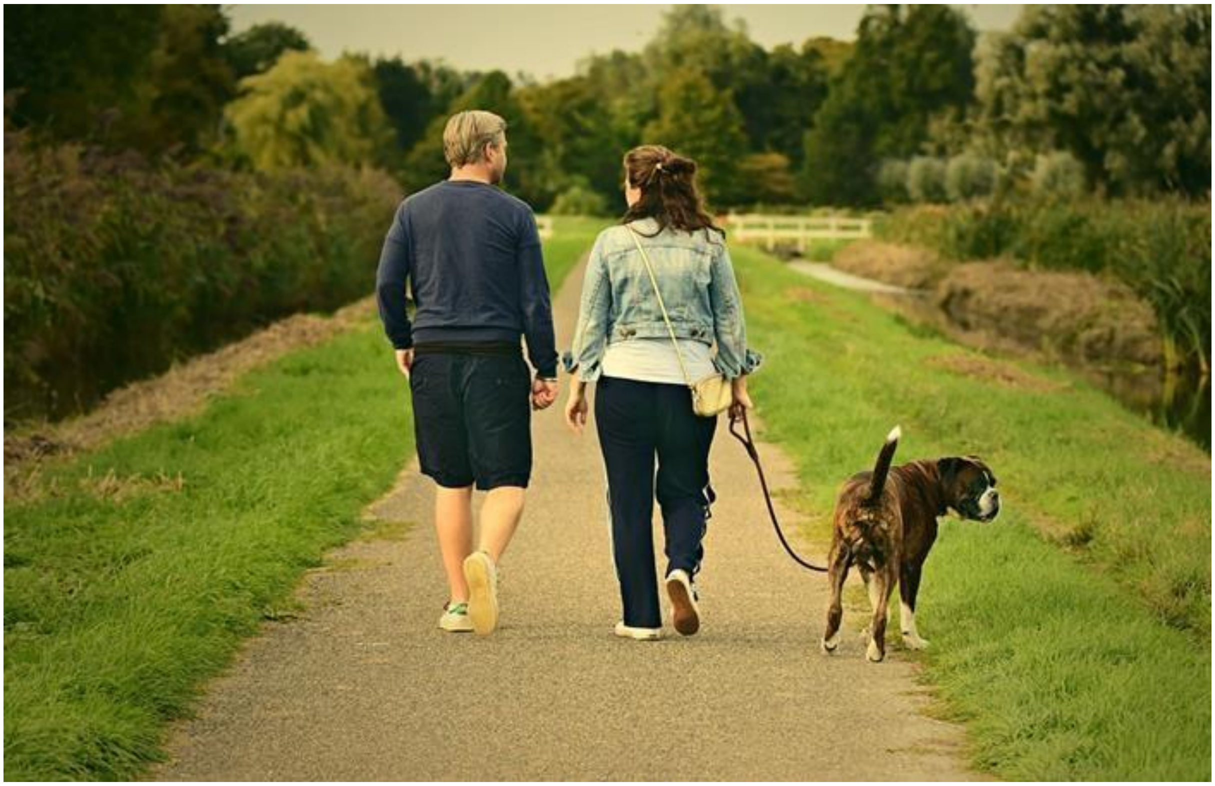


Figure 4. 5 Input Image

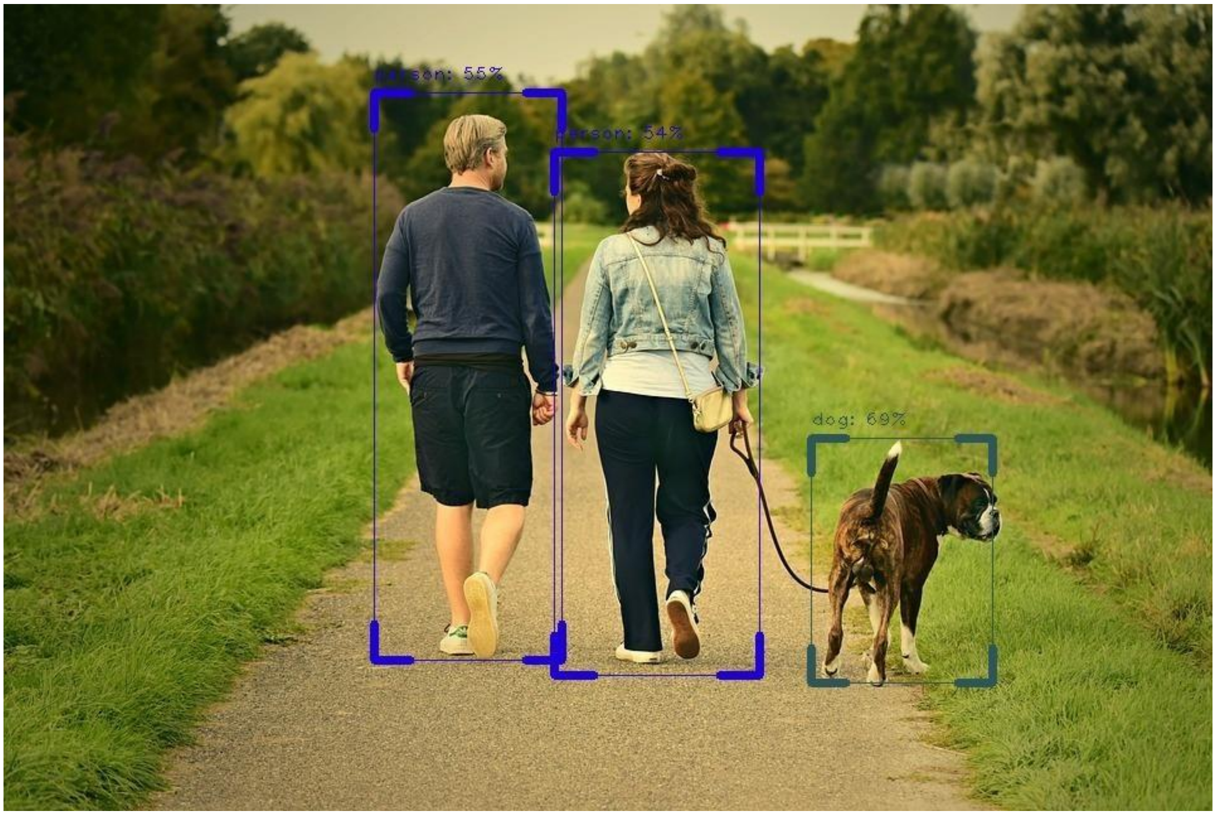


Figure 4. 6 Output Image

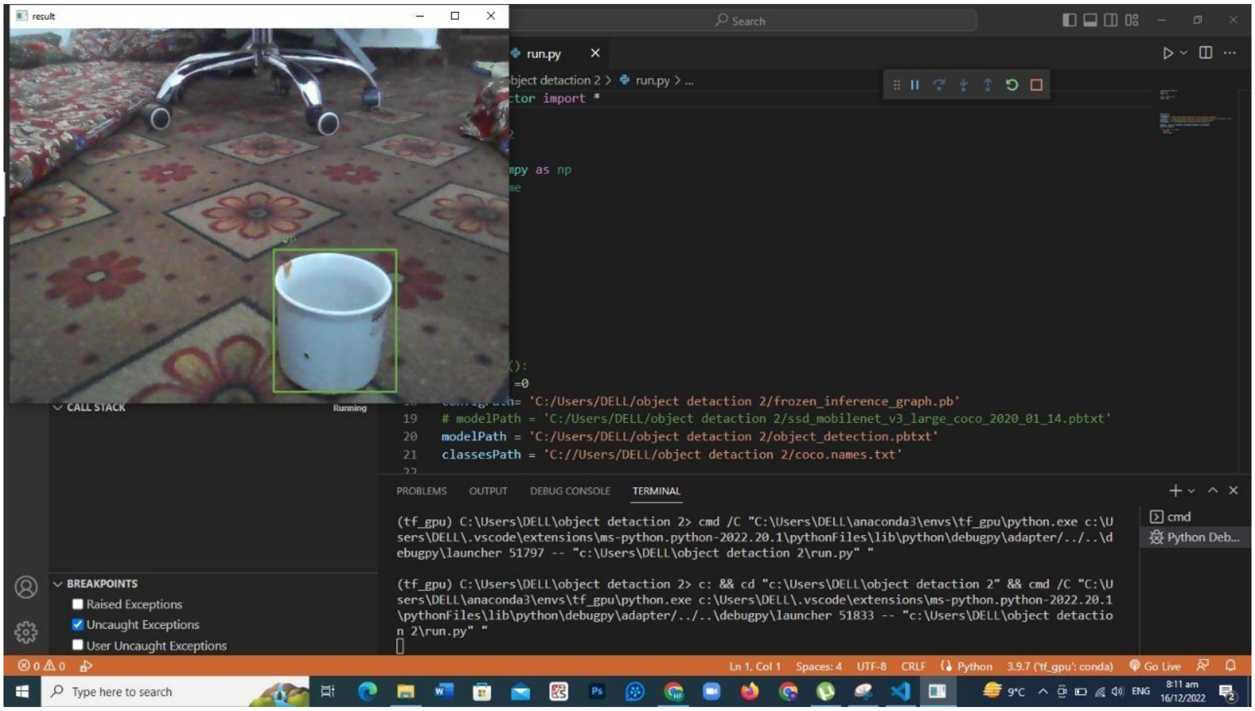
Chapter 5 Evaluation and Analysis

5.1 Evaluating the System

Every program when established needs its evaluation before deploying live in the system. Bugs can occur in any system or program so it is vital to test and evaluate it before deploying. So we have tested our algorithm with different evaluation procedures. We perform these procedures based on the input data and the output of the program and check this based on its accuracy and reliability. The evaluation process is discussed in the below chapter.

5.2 Testing the System

We evaluate the entire program and perform different tests on it. Figure 5.1 below shows our first test case in which we perform the live test in which we give the input of the cup and it predicts correctly as shown below.

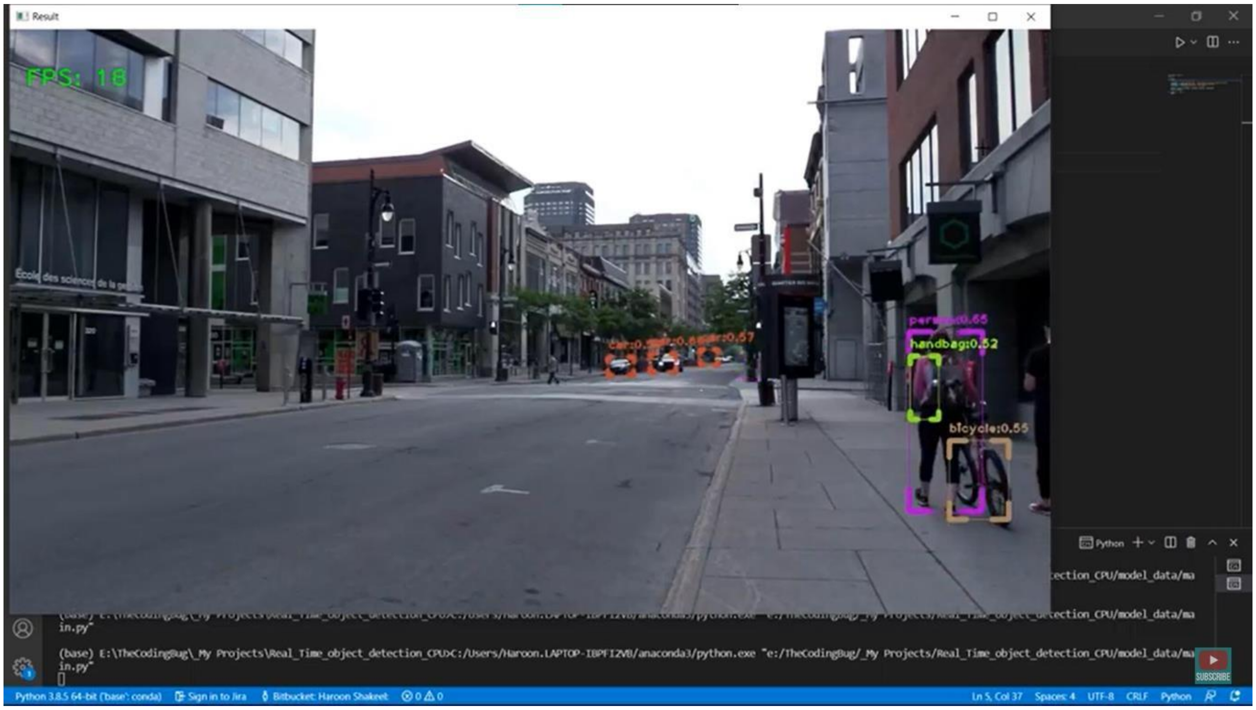


```
tor import *
mpy as np
me
():
=0
configPath= 'C:/Users/DELL/object detaction 2/frozen_inference_graph.pb'
# modelPath = 'C:/Users/DELL/object detaction 2/ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
modelPath = 'C:/Users/DELL/object detaction 2/object_detection.pbtxt'
classesPath = 'C://Users/DELL/object detaction 2/coco.names.txt'
```

```
(tf_gpu) C:\Users\DELL\object detaction 2> cmd /C "C:\Users\DELL\anaconda3\envs\tf_gpu\python.exe c:\U
sers\DELL\.vscode\extensions\ms-python.python-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\d
ebugpy\launcher 51797 -- "c:\Users\DELL\object detaction 2\run.py" "

(tf_gpu) C:\Users\DELL\object detaction 2> c: && cd "c:\Users\DELL\object detaction 2" && cmd /C "C:\U
sers\DELL\anaconda3\envs\tf_gpu\python.exe c:\Users\DELL\.vscode\extensions\ms-python.python-2022.20.1
\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher 51833 -- "c:\Users\DELL\object detactio
n 2\run.py" "
```

Figure 5. 1 Test Case 1

Below figure 5.2 shows another test procedure on our surroundings and it correctly detects all the objects in the live camera.



```
ection_CPU/model_data/ma
ection_CPU/model_data/ma
in.py"

(base) E:\TheCodingBug\_My Projects\Real_Time_object_detection_CPU>C:/Users/Haroon.LAPTOP-IBPFI2VB/anaconda3/python.exe "e:/TheCodingBug/_My Projects/Real_Time_object_detection_CPU/model_data/ma
in.py"
```

Figure 5. 2 Test Case 2

Below figure 5.3 shows another test case in which we upload a video and test it and it shows 99% correct results and detects almost every object correctly.

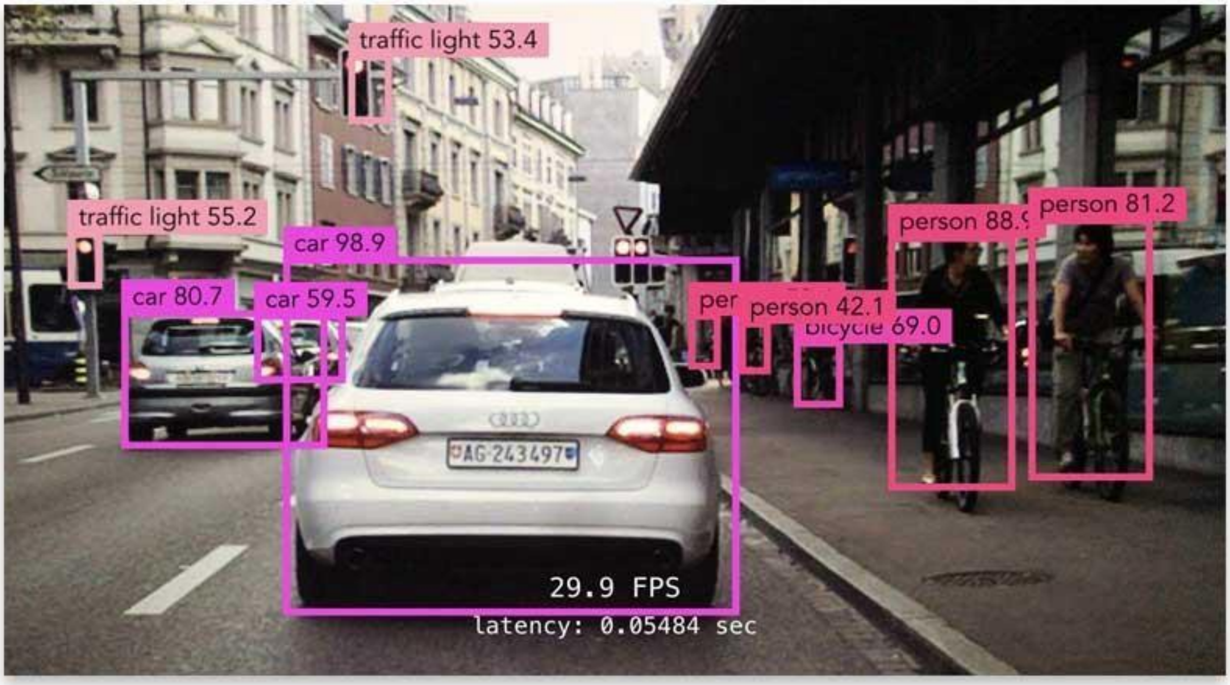


Figure 5. 3 Test Case 3

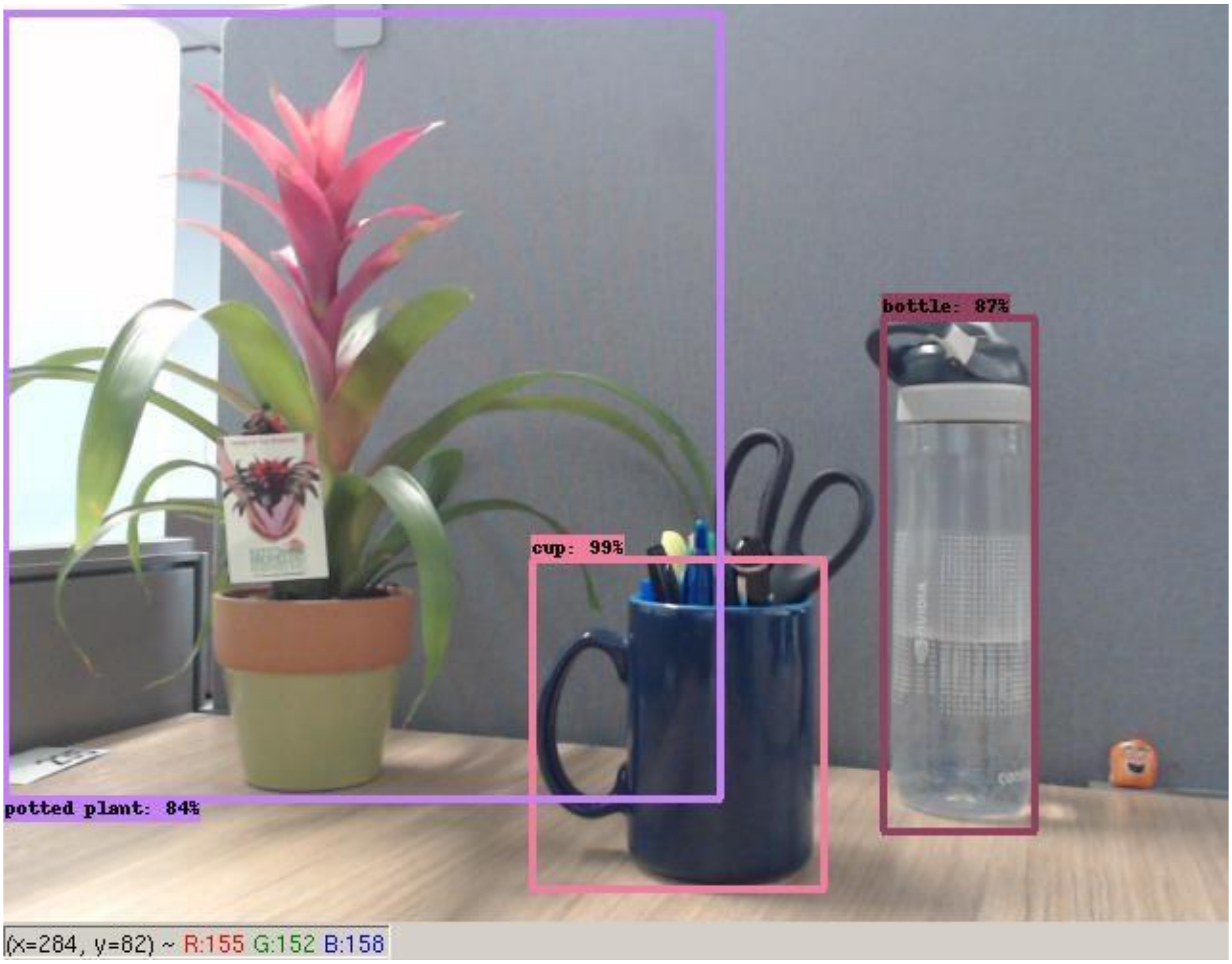


Figure 5. 4 Test Case 4

The above figure 5.4 shows another test case that also gives high accuracy.

5.3 Summary

This project aims to make such a system that can be reliable, easy to access, accurate and can predict any object in real-time on a live camera. After performing several evaluating procedures on the system we believe that this system is stable, accurate, and user-friendly. The technologies of Deep Learning and Computer Vision perform well for the system to be accurate and userfriendly. The SSD plays a key role in the system to predict the objects in real-time. The model is pre-trained and relible for the system.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

So, we can say that the system is accurate and reliable after evaluating the system. Deep learning and computer vision models perform well for this kind of system and the dataset is trained so this also helps in making the system more reliable and accurate. As we train the model more the system will perform more, also if we increase the size of the dataset it will get improve. The instrument used in the live detection will also affect the results of the system. if the camera is of lower quality then the accuracy will also be lower and if the camera is of higher quality the accuracy will be affected accordingly. Overall the system is stable and accurate.

6.2 Improvement and Future Work

As the world of technology is improving rapidly pace and with the advancement in AI technologies. We can make wonders in the system like these and make improvements in the system. If we increase the quality of data and trained it accordingly then the system will get more streamlined and improve efficiency accordingly. As more and more AI models are developed this system of multiple object detection will improve with the advanced model and also hardware affects the quality of the system result, the system processor, GPU, and camera in the future will improve more so the system can be improved more.

References

[1] A. Ekmekji, "A Review on Object Detection using Open CV Method".

[2] Amirali Abdolrashidi, "An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks".

[3] Muhammad Waqas, "Algorithm Development for moving Target detection".

[4] T. Hassner, "Multiple Object Detection using OpenCV on an Embedded Platform," The Open University of Israel.

[5] Singla, "Multi-block SSD based on small object detection for UAV railway scene surveillance,”, july 27, 2021.

[6] d. Silva, "Object Detection with Deep Learning: A Review," Department of Computer Science and Technology, Canada.