A Project Report

on

OBJECT DETECTION AND ALERT SYSTEM FOR VISUALLY IMPAIRED PEOPLE

submitted in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0525	**Ms. KURA PRAVALLIKA**
17WH1A0539	**Ms. THOTA SAHITHI**
17WH1A0559	**Ms. PALEPU PRIYOOSHA**

under the esteemed guidance of

Mr. M DYVA SUGNANA RAO

Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT B.Tech. Courses,

Accredited by NAAC with 'A' Grade)

Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

May 2021

DECLARATION

We hereby declare that the work presented in this project entitled **“OBJECT DETECTION AND ALERT SYSTEM FOR VISUALLY IMPAIRED PEOPLE**” submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women**’**, Hyderabad is an authentic record of our original work carried out under the guidance of Mr. M. Dyva Sugnana Rao, Assistant Professor, Department of CSE.

Sign. with date:

Ms. KURA PRAVALLIKA

(17WH1A0525)

Sign. with date:

Ms. THOTA SAHITHI

(17WH1A0539)

Sign. with date:

Ms. PALEPU PRIYOOSHA

(17WH1A0559)

BVRIT HYDERABAD College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT B.Tech. Courses
Accredited by NAAC with 'A' Grade)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**OBJECT DETECTION AND ALERT SYSTEM FOR VISUALLY IMPAIRED PEOPLE**" is a bonafide work carried out by Ms. KURA PRAVALLIKA (17WH1A0525) ; Ms. THOTA SAHITHI (17WH1A0539) ; Ms. PALEPU PRIYOOSHA (17WH1A0559) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. K. Srinivasa Reddy	**Mr. M. Dyva Sugnana Rao**
Professor and HoD,	**Assistant Professor**
Department of CSE	

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K. Srinivasa Reddy, Professor and HoD**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. M. Dyva Sugnana Rao, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. KURA PRAVALLIKA
(17WH1A0525)

Ms. THOTA SAHITHI
(17WH1A0539)

Ms. PALEPU PRIYOOSHA
(17WH1A0559)

ABSTRACT

Many people suffer from partial or complete blindness in this world. Though the risk of blindness due to numerous diseases has decreased by the means of medications given, it is a well-known fact that after a particular age fear of missing out on opportunities in life. The advanced technologies have proved to gain even the impossible. This project aims at helping people who are visually impaired with their navigation. The main objective revolves around implementing object detection with an alert system and embedding it into a web application that is blind-friendly. Good vision is a precious gift but unfortunately, loss of vision is becoming common nowadays. To help the blind people, the visual world has to be transformed into the audio world with the potential to inform them about objects as well as their spatial locations.

TABLE OF CONTENTS

S.No.	Topic	Page No.
1.	Introduction	1
2.	Problem Statement	3
2.1	Problem Statements	3
2.2	Specific problem statement	3
2.3	How this project solves the problem	3
3.	Literature Review	5
3.1	Existing state-of-the-art	5
3.2	Patents studied	6
4.	Methodology	6
4.1	Step by step procedure	6
4.2	YOLO	6
4.3	Block diagram	7
5.	Theoretical analysis of the proposed project	8
5.1	Requirements gathering	8
5.1.1	Software requirements	8
5.1.2	Hardware requirements	8
6.	Design	9
6.1	Introduction	9
6.2	Architecture diagram	9
7.	Implementation	11
7.1	Code	11
7.2	Testing	37
7.2.1	Testing strategies	37

7.3	Test cases	39
7.4	Dataset training screenshots	42
7.5	Input screenshots	43
7.6	Output screenshots	44
8.	Conclusion and future scope	47
9.	References	48

LIST OF FIGURES

S.No.	Topic	Page No.
1.	Object detection flow diagram	1
2.	Object detection model	1
3.	Process flow	6
4.	Yolov3 network architecture	7
5.	Architecture diagram	10
6.	Test case input 1	39
7.	Test case output 1	39
8.	Test case input 2	40
9.	Test case output 2	40
10.	Test case input 3	41
11.	Test case output 3	41
12.	Command to train the dataset	42
13.	Training dataset	42
14.	Command to run the project	43
15.	Website URL	43
16.	Website	44
17.	Image capturing	44
18.	Object detection and audio output	45
19.	Image capturing in Visual Studio Code	45
20.	Object detection and audio output in Visual Studio Code	46
21.	Detecting all the objects in the image	46

1. INTRODUCTION

Object recognition is an overall term to depict an assortment of related Computer vision tasks that include recognizing objects in digital photos. It locates the existence of an object by creating bounding boxes around it.

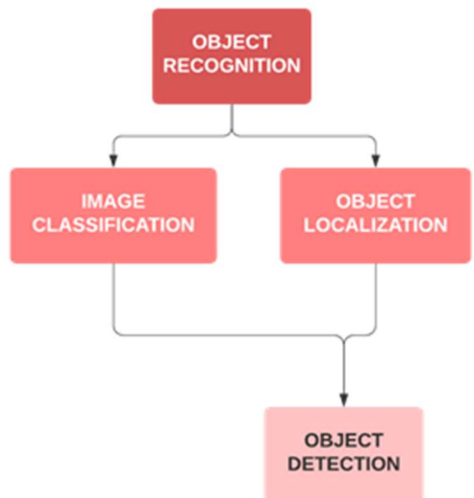

Figure 1: Object Detection Flow Diagram

A way to deal with implementing an object detection model is to initially fabricate a classifier that can classify firmly cropped pictures of an item. Figure 2 shows an instance of a comparable model, where a model is set up on a dataset of edited photos of a vehicle, a car and the model predicts the probability of an image being that vehicle.

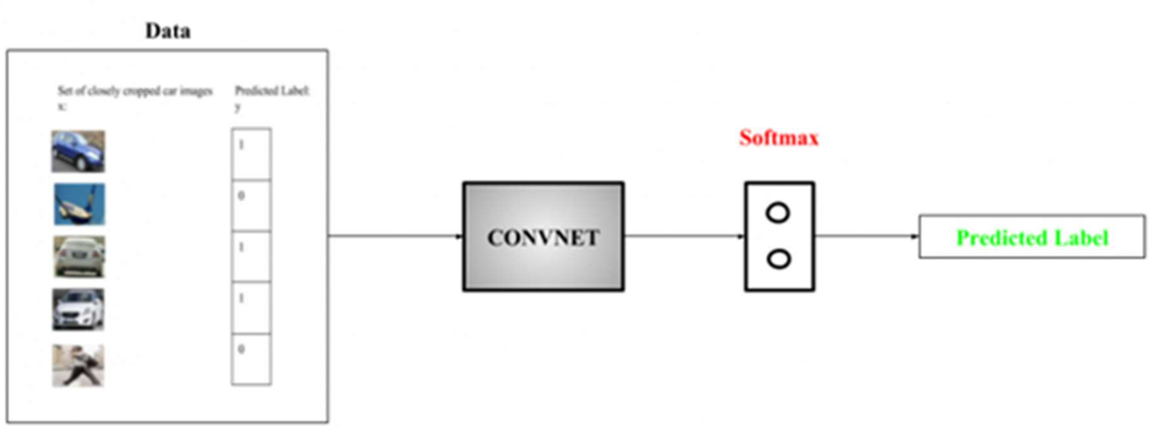

Figure 2: Object detection model

This model is now used further to detect this vehicle by following the sliding window mechanism. This mechanism is applied on the whole image. A finer model that handles the problem of anticipating exact boundary boxes by utilizing the convolutional sliding window mechanism is the YOLO algorithm. YOLO represents You Only Look Once and was created in the year 2015 by Ross Girshick, Santosh Divvala, Ali Farhadi, Joseph Redmon, and Ross Girshick. It's famous in light of the fact that it accomplishes high accuracy while running in real-time. This algorithm is popular on the grounds that it needs just one forward propagation pass across the network to put together a prediction. The algorithm dissects the picture into small grids and runs the classification of image and localization algorithm on every one of the grid cells. For instance, we have an input picture of size 256 x 256 and we use a grid of 4 x 4 .

The object detector YOLOv3 helps achieve high accuracy considering real-time performance. It is an improved version of YOLO. YOLOv3 helps in predicting the object's position using only a single neural network and only in one iteration. To achieve this, this problem is considered as a regression problem. It changes the input to class probabilities and positions.

2. PROBLEM STATEMENT

2.1. Problem Statements

Good vision is a precious gift but unfortunately loss of vision is becoming common nowadays. To help the blind people the visual world has to be transformed into the audio world with the potential to inform them about objects as well as their spatial locations.

2.2. Specific problem statement

A huge number of people suffer from partial or complete blindness in this world. Our main objective revolves around implementing object detection with an alert system and embedding it into an app which is blind friendly. Our project aims at helping people who are visually impaired or blind with their navigation.

2.3. How this project solves the problem

We will use Computer Vision technologies to implement the same. We will be able to detect and recognize objects in front of the user. We will also design and implement an alarm system to notify the user about the recognized objects using a voice assistant and give out a warning if there is any problematic situation.

3. LITERATURE REVIEW

3.1. Existing state-of-the-art

● YOLO can accurately identify objects, for instance, dustbin, within a reach of about 2-5 m ahead however the things that are beyond this reach are either not recognized or misclassified. YOLO is a real time object detection algorithm. It is the most effective and efficient object detection algorithm. It recognises what object is present in an image and where it is present. It uses a clever Convolutional neural network to detect objects in real time. An image is divided into regions and boundaries are predicted with probabilities for each region. The main advantage of YOLO is that it is extremely fast, it learns generalizable representations of objects and it scans the entire image during training and testing.

● The subsequent issue detailed by the visually impaired users is the hindering of encompassing sound by utilizing earbuds.

● The third issue announced by the visually impaired users is "data over-burden" because the software is trying to advise users of different articles at the same time. We can settle this by delaying the notifications.

3.2. Patents studied

Table 1: Patents studied

. SNo.	Existing state of art	Drawbacks in existing state of art	Overcome
1	Visual Aid (Technology Dynamics Inc)	A different output device to alert the user of the objects.	Using audio as the output, which is present in the application. No need for a different device.
2	Object Detection Device (George Brandon Foshee, Timothy Allen Zigler)	An IoT device using proximity sensors helping to identify objects and alerting the user.	IoT devices may not be available to everyone, therefore we introduce an application for this purpose.
3	Object detection system consisting of a digital camera mounted on the person's eyeglass	Cost of the final product is high and therefore not accessible to all.	Our project does not cost much in terms of hardware used. Easy integration with mobile or web apps.

4. METHODOLOGY

4.1. Step by step procedure :

1. Implement the YOLOv3 model for real time images and videos.

2. Implement code for alerting users for objects ahead.

3. Add ‘text to speech’ python script for audio output.

4. Capture images after every few seconds for detection.

5. Create a basic application for the users.

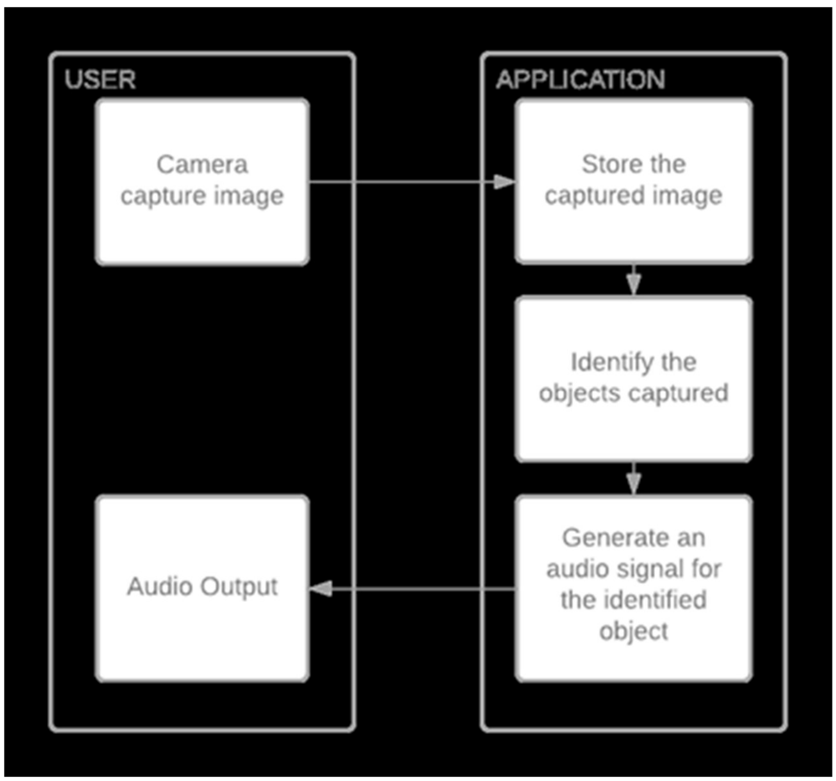

Figure 3: Process Flow

4.2. YOLO :

We will be performing real time object detection using YOLOv3.YOLO means You Only Look Once. We will detect objects with the YOLO system using pre-trained models. To apply YOLO to recordings and videos, we will write a custom command-line application in Python that utilizes a pre-prepared model to identify, localize, and classify items. It will utilize OpenCV to read the video streams, draw boxes around recognized items, name the articles alongside certainty scores, and save the marked recordings.

The following steps have been followed in this version of YOLO: YOLOv3:

1. Import cv2, matplotlib internal and utils and darknet external libraries
2. The following files need to be uploaded and neural network must be set using this information:

   a. yolov3.cfg for YOLO version 3 configuration

   b. coco.names for COCO object class

   c. yolov3.weights for pre-trained weights

3. Images are resized to our needs and converted to RGB format.
4. Non-maximal suppression must be set to 0.6. Non-maximal suppression is used by YOLO to keep only the best bounding box.
5. Set the Intersection over Union threshold value to 0.4. IOU threshold is used to select the bounding boxes with the highest detection possibility and eliminate all the bounding boxes.
6. Resized image, NMS value and IOU threshold are used to look for the objects in the image, and depending on a few factors, image is recognised with a probability.

4.3. BLOCK DIAGRAM :

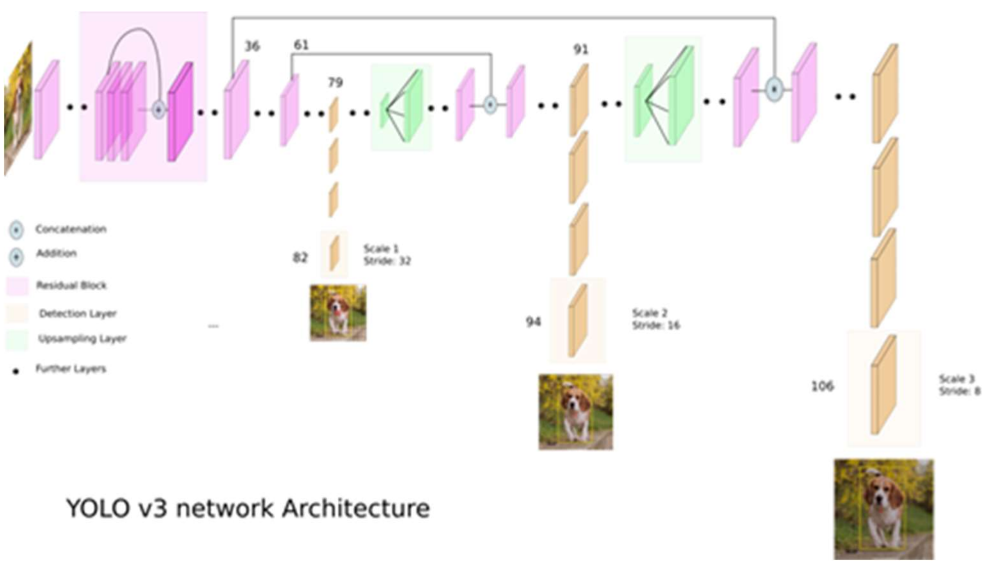

Figure 4: YOLOv3 network architecture

5. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

5.1. REQUIREMENTS GATHERING

5.1.1. SOFTWARE REQUIREMENTS:

Programming Language : Python 3.6

Graphical User Interface: HTML5, CSS3 with Bootstrap

Dataset	: Coco, yolo v3.weights file, yolov3.cfg file
Packages	: Numpy, Matplotlib, cv2, os module, gTTS, Pytorch
Framework	: Flask
Tool	: Visual Studio Code

5.1.2. HARDWARE REQUIREMENTS:

Operating System: Windows 10

Processor	: Intel Core i5
CPU Speed	: 2.30 GHz
Memory	: 8 GB (RAM)
Storage	: 1 TB

6. DESIGN

6.1. Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Once system requirements have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed, reviewed and documented. System design can be viewed from either a technical or project management perspective. From the technical point of view, design consists of four activities – architectural design, data structure design, interface design and procedural design.

6.2. Architecture Diagram

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through a network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility.

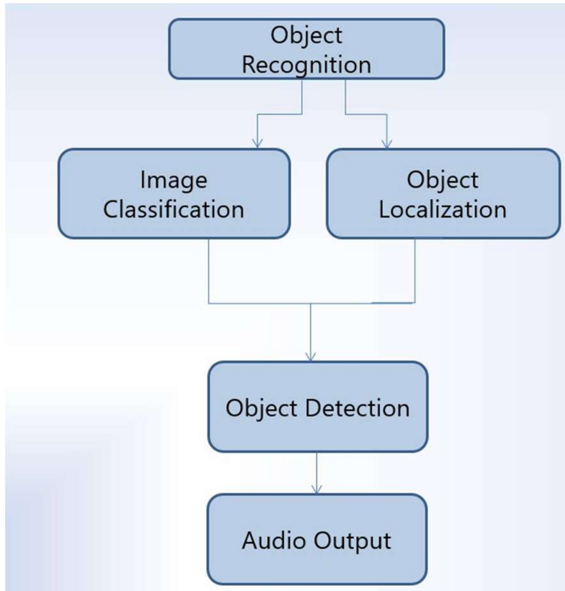

Figure 5: Architecture diagram

7. IMPLEMENTATION

7.1. CODE

➢ **Frontend: index.html**

```
<!DOCTYPE html>
<html lang="en" class="no-js">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Object Detection</title>
    <link href="https://fonts.googleapis.com/css?family=IBM+Plex+Sans:400,600" rel="stylesheet">
    <link rel="stylesheet" href= "{{ url_for('static',filename='styles/style.css') }}">
    <script src="https://unpkg.com/animejs@3.0.1/lib/anime.min.js"></script>
    <script src="https://unpkg.com/scrollreveal@4.0.0/dist/scrollreveal.min.js"></script>
    </head>
<body class="is-boxed has-animations">
    <div class="body-wrap">
        <header class="site-header">
            <div class="container">
                <div class="site-header-inner">
                    <div class="brand header-brand">
                    </div>
                </div>
            </div>
        </header>
        <main>
            <section class="hero">
```

```html
<div class="container">

    <div class="hero-copy">

        <h1 class="hero-title mt-0"><span className="text-color-primary">Object Detection and Alert System</span> for Visually Impaired People. </h1>

        <p class="hero-paragraph">This Object Detection model is one of a kind. It takes periodic input from the user to be more efficient and power saving. It detects objects in front of the camera and gives audio output for the same.</p>

        <p class="hero-paragraph">We are implementing a model for object detection using images captured periodically from the camera.</p>

        <div class="hero-cta">

            <a href="/my-link/" class="button button-primary">Using Images</a>


                </div>

            </div>

        </section>

    </main>

    <footer class="site-footer">

        <div class="container">

            <div class="site-footer-inner">

                <!-- The below ul is dummy for retaining space. -->

            </div>

        </div>

    </footer>

</div>

<script src="static/dist/js/main.min.js"></script>

</body>

</html>
```

- **Yolo implementation:**

Required files:

1. **Data - coco.names**

person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat

baseball glove
skateboard
surfboard
tennis racket
bottle
wine glass
cup
fork
knife
spoon
bowl
banana
apple
sandwich
orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
sofa
potted plant
bed
dining table
toilet
tv/monitor
laptop
mouse
remote
keyboard
cell phone
microwave
oven
toaster
sink
refrigerator
book
clock

vase
scissors
teddy bear
hair drier
toothbrush

2. **Darknet is the convolutional neural network that is used in this project. (darknet.py)**

```
import torch
import torch.nn as nn
import numpy as np
class YoloLayer(nn.Module):
    def __init__(self, anchor_mask=[], num_classes=0, anchors=[], num_anchors=1):
        super(YoloLayer, self).__init__()
        self.anchor_mask = anchor_mask
        self.num_classes = num_classes
        self.anchors = anchors
        self.num_anchors = num_anchors
        self.anchor_step = len(anchors)/num_anchors
        self.coord_scale = 1
        self.noobject_scale = 1
        self.object_scale = 5
        self.class_scale = 1
        self.thresh = 0.6
        self.stride = 32
        self.seen = 0

    def forward(self, output, nms_thresh):
        self.thresh = nms_thresh
        masked_anchors = []

        for m in self.anchor_mask:
            masked_anchors += self.anchors[m*self.anchor_step:(m+1)*self.anchor_step]

        masked_anchors = [anchor/self.stride for anchor in masked_anchors]
        boxes = get_region_boxes(output.data, self.thresh, self.num_classes, masked_anchors,
len(self.anchor_mask))

        return boxes
```

```
class Upsample(nn.Module):
    def __init__(self, stride=2):
        super(Upsample, self).__init__()
        self.stride = stride
    def forward(self, x):
        stride = self.stride
        assert(x.data.dim() == 4)
        B = x.data.size(0)
        C = x.data.size(1)
        H = x.data.size(2)
        W = x.data.size(3)
        ws = stride
        hs = stride
        x = x.view(B, C, H, 1, W, 1).expand(B, C, H, stride, W, stride).contiguous().view(B, C, H*stride,
W*stride)
        return x

#for route and shortcut
class EmptyModule(nn.Module):
    def __init__(self):
        super(EmptyModule, self).__init__()

    def forward(self, x):
        return x

# support route shortcut
class Darknet(nn.Module):
    def __init__(self, cfgfile):
        super(Darknet, self).__init__()
        self.blocks = parse_cfg(cfgfile)
        self.models = self.create_network(self.blocks) # merge conv, bn,leaky
        self.loss = self.models[len(self.models)-1]

        self.width = int(self.blocks[0]['width'])
        self.height = int(self.blocks[0]['height'])

        self.header = torch.IntTensor([0,0,0,0])
        self.seen = 0

    def forward(self, x, nms_thresh):
```

```
        ind = -2
        self.loss = None
        outputs = dict()
        out_boxes = []

        for block in self.blocks:
            ind = ind + 1
            if block['type'] == 'net':
                continue
            elif block['type'] in ['convolutional', 'upsample']:
                x = self.models[ind](x)
                outputs[ind] = x
            elif block['type'] == 'route':
                layers = block['layers'].split(',')
                layers = [int(i) if int(i) > 0 else int(i)+ind for i in layers]
                if len(layers) == 1:
                    x = outputs[layers[0]]
                    outputs[ind] = x
                elif len(layers) == 2:
                    x1 = outputs[layers[0]]
                    x2 = outputs[layers[1]]
                    x = torch.cat((x1,x2),1)
                    outputs[ind] = x
            elif block['type'] == 'shortcut':
                from_layer = int(block['from'])
                activation = block['activation']
                from_layer = from_layer if from_layer > 0 else from_layer + ind
                x1 = outputs[from_layer]
                x2 = outputs[ind-1]
                x  = x1 + x2
                outputs[ind] = x
            elif block['type'] == 'yolo':
                boxes = self.models[ind](x, nms_thresh)
                out_boxes.append(boxes)
            else:
                print('unknown type %s' % (block['type']))

        return out_boxes
    def print_network(self):
        print_cfg(self.blocks)
```

```
def create_network(self, blocks):
    models = nn.ModuleList()

    prev_filters = 3
    out_filters =[]
    prev_stride = 1
    out_strides = []
    conv_id = 0
    for block in blocks:
        if block['type'] == 'net':
            prev_filters = int(block['channels'])
            continue
        elif block['type'] == 'convolutional':
            conv_id = conv_id + 1
            batch_normalize = int(block['batch_normalize'])
            filters = int(block['filters'])
            kernel_size = int(block['size'])
            stride = int(block['stride'])
            is_pad = int(block['pad'])
            pad = (kernel_size-1)//2 if is_pad else 0
            activation = block['activation']
            model = nn.Sequential()
            if batch_normalize:
                model.add_module('conv{0}'.format(conv_id), nn.Conv2d(prev_filters, filters, kernel_size,
stride, pad, bias=False))
                model.add_module('bn{0}'.format(conv_id), nn.BatchNorm2d(filters))
            else:
                model.add_module('conv{0}'.format(conv_id), nn.Conv2d(prev_filters, filters, kernel_size,
stride, pad))
            if activation == 'leaky':
                model.add_module('leaky{0}'.format(conv_id), nn.LeakyReLU(0.1, inplace=True))
            prev_filters = filters
            out_filters.append(prev_filters)
            prev_stride = stride * prev_stride
            out_strides.append(prev_stride)
            models.append(model)
        elif block['type'] == 'upsample':
            stride = int(block['stride'])
            out_filters.append(prev_filters)
```

```
            prev_stride = prev_stride // stride
            out_strides.append(prev_stride)
            models.append(Upsample(stride))
        elif block['type'] == 'route':
            layers = block['layers'].split(',')
            ind = len(models)
            layers = [int(i) if int(i) > 0 else int(i)+ind for i in layers]
            if len(layers) == 1:
                prev_filters = out_filters[layers[0]]
                prev_stride = out_strides[layers[0]]
            elif len(layers) == 2:
                assert(layers[0] == ind - 1)
                prev_filters = out_filters[layers[0]] + out_filters[layers[1]]
                prev_stride = out_strides[layers[0]]
            out_filters.append(prev_filters)
            out_strides.append(prev_stride)
            models.append(EmptyModule())
        elif block['type'] == 'shortcut':
            ind = len(models)
            prev_filters = out_filters[ind-1]
            out_filters.append(prev_filters)
            prev_stride = out_strides[ind-1]
            out_strides.append(prev_stride)
            models.append(EmptyModule())
        elif block['type'] == 'yolo':
            yolo_layer = YoloLayer()
            anchors = block['anchors'].split(',')
            anchor_mask = block['mask'].split(',')
            yolo_layer.anchor_mask = [int(i) for i in anchor_mask]
            yolo_layer.anchors = [float(i) for i in anchors]
            yolo_layer.num_classes = int(block['classes'])
            yolo_layer.num_anchors = int(block['num'])
            yolo_layer.anchor_step = len(yolo_layer.anchors)//yolo_layer.num_anchors
            yolo_layer.stride = prev_stride
            out_filters.append(prev_filters)
            out_strides.append(prev_stride)
            models.append(yolo_layer)
        else:
            print('unknown type %s' % (block['type']))
    return models
```

```
def load_weights(self, weightfile):
    print()
    fp = open(weightfile, 'rb')
    header = np.fromfile(fp, count=5, dtype=np.int32)
    self.header = torch.from_numpy(header)
    self.seen = self.header[3]
    buf = np.fromfile(fp, dtype = np.float32)
    fp.close()

    start = 0
    ind = -2
    counter = 3
    for block in self.blocks:
        if start >= buf.size:
            break
        ind = ind + 1
        if block['type'] == 'net':
            continue
        elif block['type'] == 'convolutional':
            model = self.models[ind]
            batch_normalize = int(block['batch_normalize'])
            if batch_normalize:
                start = load_conv_bn(buf, start, model[0], model[1])
            else:
                start = load_conv(buf, start, model[0])
        elif block['type'] == 'upsample':
            pass
        elif block['type'] == 'route':
            pass
        elif block['type'] == 'shortcut':
            pass
        elif block['type'] == 'yolo':
            pass
        else:
            print('unknown type %s' % (block['type']))

        percent_comp = (counter / len(self.blocks)) * 100
```

```
        print('Loading weights. Please Wait...{:.2f}% Complete'.format(percent_comp), end = '\r', flush =
True)

        counter += 1
def convert2cpu(gpu_matrix):
    return torch.FloatTensor(gpu_matrix.size()).copy_(gpu_matrix)

def convert2cpu_long(gpu_matrix):
    return torch.LongTensor(gpu_matrix.size()).copy_(gpu_matrix)

def get_region_boxes(output, conf_thresh, num_classes, anchors, num_anchors, only_objectness = 1,
validation = False):
    anchor_step = len(anchors)//num_anchors
    if output.dim() == 3:
        output = output.unsqueeze(0)
    batch = output.size(0)
    assert(output.size(1) == (5+num_classes)*num_anchors)
    h = output.size(2)
    w = output.size(3)

    all_boxes = []
    output = output.view(batch*num_anchors, 5+num_classes,
h*w).transpose(0,1).contiguous().view(5+num_classes, batch*num_anchors*h*w)

    grid_x = torch.linspace(0, w-1, w).repeat(h,1).repeat(batch*num_anchors, 1,
1).view(batch*num_anchors*h*w).type_as(output) #cuda()
    grid_y = torch.linspace(0, h-1, h).repeat(w,1).t().repeat(batch*num_anchors, 1,
1).view(batch*num_anchors*h*w).type_as(output) #cuda()
    xs = torch.sigmoid(output[0]) + grid_x
    ys = torch.sigmoid(output[1]) + grid_y

    anchor_w = torch.Tensor(anchors).view(num_anchors, anchor_step).index_select(1,
torch.LongTensor([0]))
    anchor_h = torch.Tensor(anchors).view(num_anchors, anchor_step).index_select(1,
torch.LongTensor([1]))
    anchor_w = anchor_w.repeat(batch, 1).repeat(1, 1,
h*w).view(batch*num_anchors*h*w).type_as(output) #cuda()
    anchor_h = anchor_h.repeat(batch, 1).repeat(1, 1,
h*w).view(batch*num_anchors*h*w).type_as(output) #cuda()
    ws = torch.exp(output[2]) * anchor_w
```

```
hs = torch.exp(output[3]) * anchor_h

det_confs = torch.sigmoid(output[4])
cls_confs = torch.nn.Softmax(dim=1)(output[5:5+num_classes].transpose(0,1)).detach()
cls_max_confs, cls_max_ids = torch.max(cls_confs, 1)
cls_max_confs = cls_max_confs.view(-1)
cls_max_ids = cls_max_ids.view(-1)

sz_hw = h*w
sz_hwa = sz_hw*num_anchors
det_confs = convert2cpu(det_confs)
cls_max_confs = convert2cpu(cls_max_confs)
cls_max_ids = convert2cpu_long(cls_max_ids)
xs = convert2cpu(xs)
ys = convert2cpu(ys)
ws = convert2cpu(ws)
hs = convert2cpu(hs)
if validation:
    cls_confs = convert2cpu(cls_confs.view(-1, num_classes))

for b in range(batch):
    boxes = []
    for cy in range(h):
        for cx in range(w):
            for i in range(num_anchors):
                ind = b*sz_hwa + i*sz_hw + cy*w + cx
                det_conf =  det_confs[ind]
                if only_objectness:
                    conf =  det_confs[ind]
                else:
                    conf = det_confs[ind] * cls_max_confs[ind]

                if conf > conf_thresh:
                    bcx = xs[ind]
                    bcy = ys[ind]
                    bw = ws[ind]
                    bh = hs[ind]
                    cls_max_conf = cls_max_confs[ind]
                    cls_max_id = cls_max_ids[ind]
                    box = [bcx/w, bcy/h, bw/w, bh/h, det_conf, cls_max_conf, cls_max_id]
```

```
                if (not only_objectness) and validation:
                    for c in range(num_classes):
                        tmp_conf = cls_confs[ind][c]
                        if c != cls_max_id and det_confs[ind]*tmp_conf > conf_thresh:
                            box.append(tmp_conf)
                            box.append(c)
                boxes.append(box)
        all_boxes.append(boxes)

    return all_boxes

def parse_cfg(cfgfile):
    blocks = []
    fp = open(cfgfile, 'r')
    block =  None
    line = fp.readline()
    while line != '':
        line = line.rstrip()
        if line == '' or line[0] == '#':
            line = fp.readline()
            continue
        elif line[0] == '[':
            if block:
                blocks.append(block)
            block = dict()
            block['type'] = line.lstrip('[').rstrip(']')
            # set default value
            if block['type'] == 'convolutional':
                block['batch_normalize'] = 0
        else:
            key,value = line.split('=')
            key = key.strip()
            if key == 'type':
                key = '_type'
            value = value.strip()
            block[key] = value
        line = fp.readline()

    if block:
        blocks.append(block)
```

```
    fp.close()
    return blocks

def print_cfg(blocks):
    print('layer     filters    size              input                output');
    prev_width = 416
    prev_height = 416
    prev_filters = 3
    out_filters =[]
    out_widths =[]
    out_heights =[]
    ind = -2
    for block in blocks:
        ind = ind + 1
        if block['type'] == 'net':
            prev_width = int(block['width'])
            prev_height = int(block['height'])
            continue
        elif block['type'] == 'convolutional':
            filters = int(block['filters'])
            kernel_size = int(block['size'])
            stride = int(block['stride'])
            is_pad = int(block['pad'])
            pad = (kernel_size-1)//2 if is_pad else 0
            width = (prev_width + 2*pad - kernel_size)//stride + 1
            height = (prev_height + 2*pad - kernel_size)//stride + 1
            print('%5d %-6s %4d  %d x %d / %d   %3d x %3d x%4d   ->   %3d x %3d x%4d' % (ind, 'conv',
filters, kernel_size, kernel_size, stride, prev_width, prev_height, prev_filters, width, height, filters))
            prev_width = width
            prev_height = height
            prev_filters = filters
            out_widths.append(prev_width)
            out_heights.append(prev_height)
            out_filters.append(prev_filters)
        elif block['type'] == 'upsample':
            stride = int(block['stride'])
            filters = prev_filters
            width = prev_width*stride
            height = prev_height*stride
```

```
            print('%5d %-6s          * %d   %3d x %3d x%4d   ->   %3d x %3d x%4d' % (ind, 'upsample', stride,
prev_width, prev_height, prev_filters, width, height, filters))
            prev_width = width
            prev_height = height
            prev_filters = filters
            out_widths.append(prev_width)
            out_heights.append(prev_height)
            out_filters.append(prev_filters)
        elif block['type'] == 'route':
            layers = block['layers'].split(',')
            layers = [int(i) if int(i) > 0 else int(i)+ind for i in layers]
            if len(layers) == 1:
                print('%5d %-6s %d' % (ind, 'route', layers[0]))
                prev_width = out_widths[layers[0]]
                prev_height = out_heights[layers[0]]
                prev_filters = out_filters[layers[0]]
            elif len(layers) == 2:
                print('%5d %-6s %d %d' % (ind, 'route', layers[0], layers[1]))
                prev_width = out_widths[layers[0]]
                prev_height = out_heights[layers[0]]
                assert(prev_width == out_widths[layers[1]])
                assert(prev_height == out_heights[layers[1]])
                prev_filters = out_filters[layers[0]] + out_filters[layers[1]]
            out_widths.append(prev_width)
            out_heights.append(prev_height)
            out_filters.append(prev_filters)
        elif block['type'] in ['region', 'yolo']:
            print('%5d %-6s' % (ind, 'detection'))
            out_widths.append(prev_width)
            out_heights.append(prev_height)
            out_filters.append(prev_filters)
        elif block['type'] == 'shortcut':
            from_id = int(block['from'])
            from_id = from_id if from_id > 0 else from_id+ind
            print('%5d %-6s %d' % (ind, 'shortcut', from_id))
            prev_width = out_widths[from_id]
            prev_height = out_heights[from_id]
            prev_filters = out_filters[from_id]
            out_widths.append(prev_width)
            out_heights.append(prev_height)
```

```
            out_filters.append(prev_filters)
        else:
            print('unknown type %s' % (block['type']))

def load_conv(buf, start, conv_model):
    num_w = conv_model.weight.numel()
    num_b = conv_model.bias.numel()
    conv_model.bias.data.copy_(torch.from_numpy(buf[start:start+num_b]));   start = start + num_b

conv_model.weight.data.copy_(torch.from_numpy(buf[start:start+num_w]).view_as(conv_model.weigh
t.data)); start = start + num_w
    return start

def load_conv_bn(buf, start, conv_model, bn_model):
    num_w = conv_model.weight.numel()
    num_b = bn_model.bias.numel()
    bn_model.bias.data.copy_(torch.from_numpy(buf[start:start+num_b]));     start = start + num_b
    bn_model.weight.data.copy_(torch.from_numpy(buf[start:start+num_b]));   start = start + num_b
    bn_model.running_mean.copy_(torch.from_numpy(buf[start:start+num_b]));  start = start + num_b
    bn_model.running_var.copy_(torch.from_numpy(buf[start:start+num_b]));   start = start + num_b

conv_model.weight.data.copy_(torch.from_numpy(buf[start:start+num_w]).view_as(conv_model.weigh
t.data)); start = start + num_w
    return start
```

3. **Utils is an external package that contains the image processing functions used in this project. (utils.py)**

```
import time
import torch
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from gtts import gTTS
import os


def boxes_iou(box1, box2):

    # Get the Width and Height of each bounding box
```

```
width_box1 = box1[2]
height_box1 = box1[3]
width_box2 = box2[2]
height_box2 = box2[3]

# Calculate the area of the each bounding box
area_box1 = width_box1 * height_box1
area_box2 = width_box2 * height_box2

# Find the vertical edges of the union of the two bounding boxes
mx = min(box1[0] - width_box1/2.0, box2[0] - width_box2/2.0)
Mx = max(box1[0] + width_box1/2.0, box2[0] + width_box2/2.0)

# Calculate the width of the union of the two bounding boxes
union_width = Mx - mx

# Find the horizontal edges of the union of the two bounding boxes
my = min(box1[1] - height_box1/2.0, box2[1] - height_box2/2.0)
My = max(box1[1] + height_box1/2.0, box2[1] + height_box2/2.0)

# Calculate the height of the union of the two bounding boxes
union_height = My - my

# Calculate the width and height of the area of intersection of the two bounding boxes
intersection_width = width_box1 + width_box2 - union_width
intersection_height = height_box1 + height_box2 - union_height

# If the the boxes don't overlap then their IOU is zero
if intersection_width <= 0 or intersection_height <= 0:
    return 0.0

# Calculate the area of intersection of the two bounding boxes
intersection_area = intersection_width * intersection_height

# Calculate the area of the union of the two bounding boxes
union_area = area_box1 + area_box2 - intersection_area

# Calculate the IOU
iou = intersection_area/union_area
```

```
    return iou


def nms(boxes, iou_thresh):

    # If there are no bounding boxes do nothing
    if len(boxes) == 0:
        return boxes

    # Create a PyTorch Tensor to keep track of the detection confidence
    # of each predicted bounding box
    det_confs = torch.zeros(len(boxes))

    # Get the detection confidence of each predicted bounding box
    for i in range(len(boxes)):
        det_confs[i] = boxes[i][4]

    # Sort the indices of the bounding boxes by detection confidence value in descending order.
    # We ignore the first returned element since we are only interested in the sorted indices
    _,sortIds = torch.sort(det_confs, descending = True)

    # Create an empty list to hold the best bounding boxes after
    # Non-Maximal Suppression (NMS) is performed
    best_boxes = []

    # Perform Non-Maximal Suppression
    for i in range(len(boxes)):

        # Get the bounding box with the highest detection confidence first
        box_i = boxes[sortIds[i]]

        # Check that the detection confidence is not zero
        if box_i[4] > 0:

            # Save the bounding box
            best_boxes.append(box_i)

            # Go through the rest of the bounding boxes in the list and calculate their IOU with
            # respect to the previous selected box_i.
            for j in range(i + 1, len(boxes)):
```

```
            box_j = boxes[sortIds[j]]

            # If the IOU of box_i and box_j is higher than the given IOU threshold set
            # box_j's detection confidence to zero.
            if boxes_iou(box_i, box_j) > iou_thresh:
                box_j[4] = 0

    return best_boxes


def detect_objects(model, img, iou_thresh, nms_thresh):

    # Start the time. This is done to calculate how long the detection takes.
    start = time.time()

    # Set the model to evaluation mode.
    model.eval()

    # Convert the image from a NumPy ndarray to a PyTorch Tensor of the correct shape.
    # The image is transposed, then converted to a FloatTensor of dtype float32, then
    # Normalized to values between 0 and 1, and finally unsqueezed to have the correct
    # shape of 1 x 3 x 416 x 416
    img = torch.from_numpy(img.transpose(2,0,1)).float().div(255.0).unsqueeze(0)

    # Feed the image to the neural network with the corresponding NMS threshold.
    # The first step in NMS is to remove all bounding boxes that have a very low
    # probability of detection. All predicted bounding boxes with a value less than
    # the given NMS threshold will be removed.
    list_boxes = model(img, nms_thresh)

    # Make a new list with all the bounding boxes returned by the neural network
    boxes = list_boxes[0][0] + list_boxes[1][0] + list_boxes[2][0]

    # Perform the second step of NMS on the bounding boxes returned by the neural network.
    # In this step, we only keep the best bounding boxes by eliminating all the bounding boxes
    # whose IOU value is higher than the given IOU threshold
    boxes = nms(boxes, iou_thresh)

    # Stop the time.
    finish = time.time()
```

```
    # # Print the time it took to detect objects
    # print('\n\nIt took {:.3f}'.format(finish - start), 'seconds to detect the objects in the image.\n')
    #
    # # Print the number of objects detected
    # print('Number of Objects Detected:', len(boxes), '\n')

    return boxes


def load_class_names(namesfile):

    # Create an empty list to hold the object classes
    class_names = []

    # Open the file containing the COCO object classes in read-only mode
    with open(namesfile, 'r') as fp:

        # The coco.names file contains only one object class per line.
        # Read the file line by line and save all the lines in a list.
        lines = fp.readlines()

    # Get the object class names
    for line in lines:

        # Make a copy of each line with any trailing whitespace removed
        line = line.rstrip()

        # Save the object class name into class_names
        class_names.append(line)

    return class_names


def print_objects(boxes, class_names):
    for i in range(len(boxes)):
        box = boxes[i]
        if len(box) >= 7 and class_names:
            cls_conf = box[5]
            cls_id = box[6]
```

```
        print(class_names[cls_id])
        myobj = gTTS(text=class_names[cls_id], lang='en', slow=False)
        myobj.save("welcome.mp3")
        os.system("start welcome.mp3")
        time.sleep(2)


def plot_boxes(img, boxes, class_names, plot_labels, color = None):

    # Define a tensor used to set the colors of the bounding boxes
    colors = torch.FloatTensor([[1,0,1],[0,0,1],[0,1,1],[0,1,0],[1,1,0],[1,0,0]])

    # Define a function to set the colors of the bounding boxes
    def get_color(c, x, max_val):
        ratio = float(x) / max_val * 5
        i = int(np.floor(ratio))
        j = int(np.ceil(ratio))

        ratio = ratio - i
        r = (1 - ratio) * colors[i][c] + ratio * colors[j][c]

        return int(r * 255)

    # Get the width and height of the image
    width = img.shape[1]
    height = img.shape[0]

    # Create a figure and plot the image
    fig, a = plt.subplots(1,1)
    a.imshow(img)

    # Plot the bounding boxes and corresponding labels on top of the image
    for i in range(len(boxes)):

        # Get the ith bounding box
        box = boxes[i]

        # Get the (x,y) pixel coordinates of the lower-left and lower-right corners
        # of the bounding box relative to the size of the image.
        x1 = int(np.around((box[0] - box[2]/2.0) * width))
```

```
y1 = int(np.around((box[1] - box[3]/2.0) * height))
x2 = int(np.around((box[0] + box[2]/2.0) * width))
y2 = int(np.around((box[1] + box[3]/2.0) * height))

# Set the default rgb value to red
rgb = (1, 0, 0)

# Use the same color to plot the bounding boxes of the same object class
if len(box) >= 7 and class_names:
    cls_conf = box[5]
    cls_id = box[6]
    classes = len(class_names)
    offset = cls_id * 123457 % classes
    red   = get_color(2, offset, classes) / 255
    green = get_color(1, offset, classes) / 255
    blue  = get_color(0, offset, classes) / 255

    # If a color is given then set rgb to the given color instead
    if color is None:
        rgb = (red, green, blue)
    else:
        rgb = color

# Calculate the width and height of the bounding box relative to the size of the image.
width_x = x2 - x1
width_y = y1 - y2

# Set the position and size of the bounding box. (x1, y2) is the pixel coordinate of the
# lower-left corner of the bounding box relative to the size of the image.
rect = patches.Rectangle((x1, y2),
                         width_x, width_y,
                         linewidth = 2,
                         edgecolor = rgb,
                         facecolor = 'none')

# Draw the bounding box on top of the image
a.add_patch(rect)

# If plot_labels = True then plot the corresponding label
if plot_labels:
```

```
        # Create a string with the object class name and the corresponding object class probability
        conf_tx = class_names[cls_id] + ': {:.1f}'.format(cls_conf)

        # Define x and y offsets for the labels
        lxc = (img.shape[1] * 0.266) / 100
        lyc = (img.shape[0] * 1.180) / 100

        # Draw the labels on top of the image
        a.text(x1 + lxc, y1 - lyc, conf_tx, fontsize = 24, color = 'k',
               bbox = dict(facecolor = rgb, edgecolor = rgb, alpha = 0.8))

    plt.show()
```

4. **Yolov3.cfg is the configuration file which is passed into the darknet to configure it to work as yolo.**

➢ **Backend:** This includes backend and yolo implementation code.

Backend.py

```
# from yolo import ImageDetection
import cv2
import matplotlib.pyplot as plt

from utils import *
from darknet import Darknet

from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def home():
    return render_template("index.html")

#YOLO Implementation
@app.route('/my-link/')
def my_link():
    # Number of frames to throw away while the camera adjusts to light levels
```

```
ramp_frames = 30

# Now we can initialize the camera capture object with the cv2.VideoCapture class.
# All it needs is the index to a camera port.
camera = cv2.VideoCapture(0)

# Captures a single image from the camera and returns it in PIL format
def get_image():
    # read is the easiest way to get a full image out of a VideoCapture object.
    retval, im = camera.read()
    return im


# Set the location and name of the cfg file
cfg_file = './cfg/yolov3.cfg'

# Set the location and name of the pre-trained weights file
weight_file = './weights/yolov3.weights'

# Set the location and name of the COCO object classes file
namesfile = 'data/coco.names'

# Load the network architecture
m = Darknet(cfg_file)

# Load the pre-trained weights
m.load_weights(weight_file)

# Load the COCO object classes
class_names = load_class_names(namesfile)

# # Print the neural network used in YOLOv3
# m.print_network()

# Set the default figure size
plt.rcParams['figure.figsize'] = [24.0, 14.0]

# Load the image
img = cv2.imread('./images/dog.jpg')
```

```
# Convert the image to RGB
original_image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# We resize the image to the input width and height of the first layer of the network.
resized_image = cv2.resize(original_image, (m.width, m.height))

i = 5

while i > 1:
    i-=1
    # Ramp the camera - these frames will be discarded and are only used to allow v4l2
    # to adjust light levels, if necessary

    camera = cv2.VideoCapture(0)
    for i in range(ramp_frames):
        temp = get_image()
    print("Taking image...")
    # Take the actual image we want to keep
    camera_capture = get_image()
    file = "./test_image.png"
    # A nice feature of the imwrite method is that it will automatically choose the
    # correct format based on the file extension you provide. Convenient!
    cv2.imwrite(file, camera_capture)

    # You'll want to release the camera, otherwise you won't be able to create a new
    # capture object until your script exits
    del camera

    # Set the default figure size
    plt.rcParams['figure.figsize'] = [24.0, 14.0]

    # Load the image
    img = cv2.imread('./test_image.png')

    # Convert the image to RGB
    original_image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # We resize the image to the input width and height of the first layer of the network.
    resized_image = cv2.resize(original_image, (m.width, m.height))
```

```
    # Set the IOU threshold. Default value is 0.4
    iou_thresh = 0.4

    # Set the NMS threshold. Default value is 0.6
    nms_thresh = 0.6

    # Detect objects in the image
    boxes = detect_objects(m, resized_image, iou_thresh, nms_thresh)

    # Print the objects found and the confidence level
    print_objects(boxes, class_names)

if __name__ == "__main__":
app.run()
```

7.2. TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There are basically two types of testing approaches.

One is Black-Box testing – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is White-Box testing – knowing the internal workings of the product ,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package.The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

7.2.1 Testing Strategies

Testing is a set of activities that can be planned in advance and conducted systematically. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements a specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test techniques that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

UNIT TESTING:

This testing method considers a module as a single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combinations are pre-calculated and are generated by the module.

SYSTEM TESTING:

Here all the pre-tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchrony with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules at once and then generate test combinations of test paths throughout the system to make sure that no path is making its way into chaos.

INTEGRATED TESTING:

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce more than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and build a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an endless loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top-down integration, bottom-up integration, and regression testing.

REGRESSION TESTING:

Each time a new module is added as a part of integration as the software changes. Regression testing is an actual that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing may be conducted manually by executing a subset of all test cases or using automated capture playback tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

A representative sample of tests that will exercise all software functions. Additional tests that focus on software functions that are likely to be affected by the change.

7.3 TEST CASES

Unit testing strategy is used in this application for testing.

[A]

Figure 6: Test case input 1

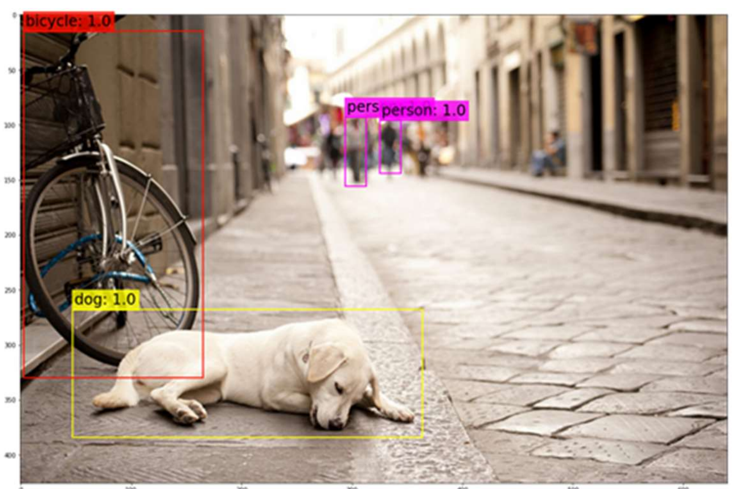

Figure 7: Test case output 1

1. dog: 0.999024
2. bicycle: 0.999822
3. person: 1.000000
4. person: 1.000000

[B]

Figure 8: Test case input 2

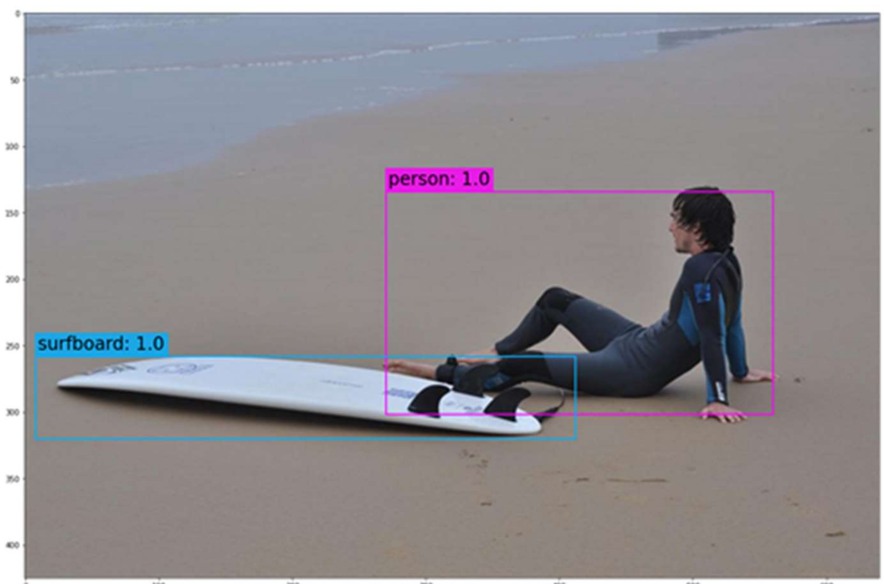

Figure 9: Test case output 2

1. person: 1.000000

2. surfboard: 0.994814

[C]

Figure 10: Test case input 3

Figure 11: Test case output 3

1. tv/monitor: 0.999921
2. bottle: 0.994491
3. vase: 0.999943

7.4 DATASET TRAINING SCREENSHOTS

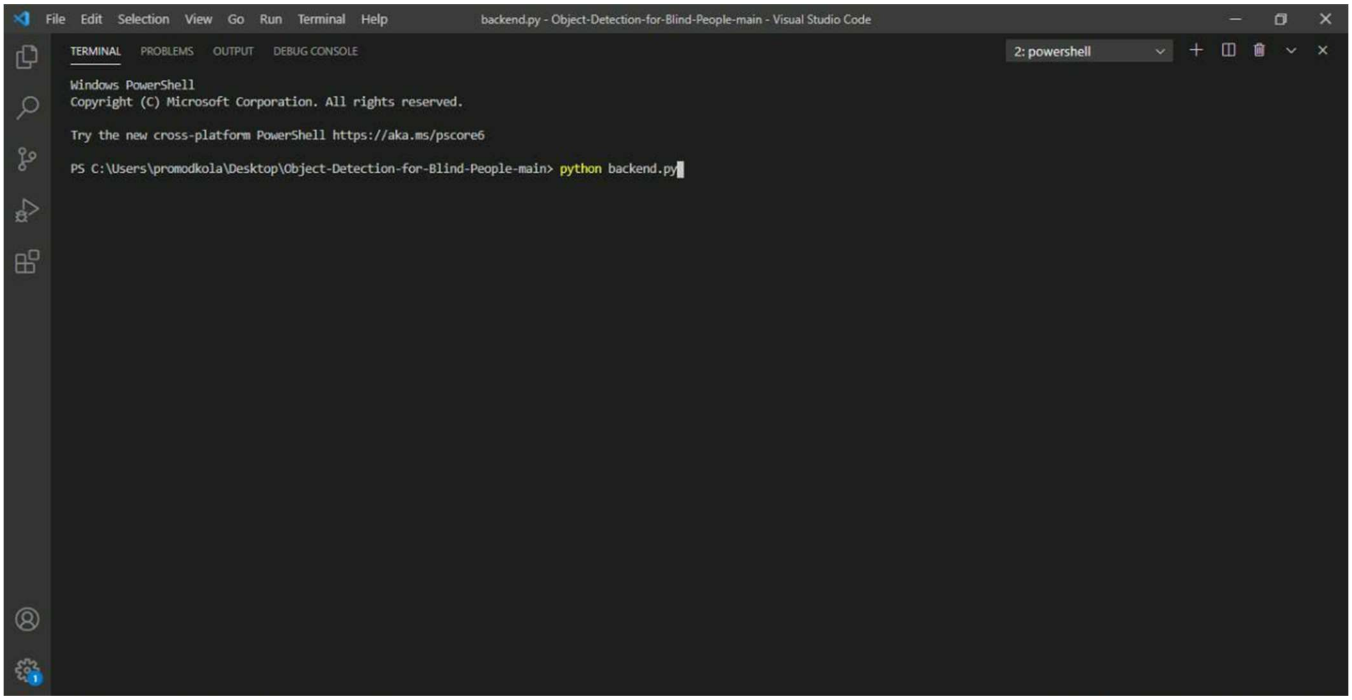

Figure 12: Command to train the dataset

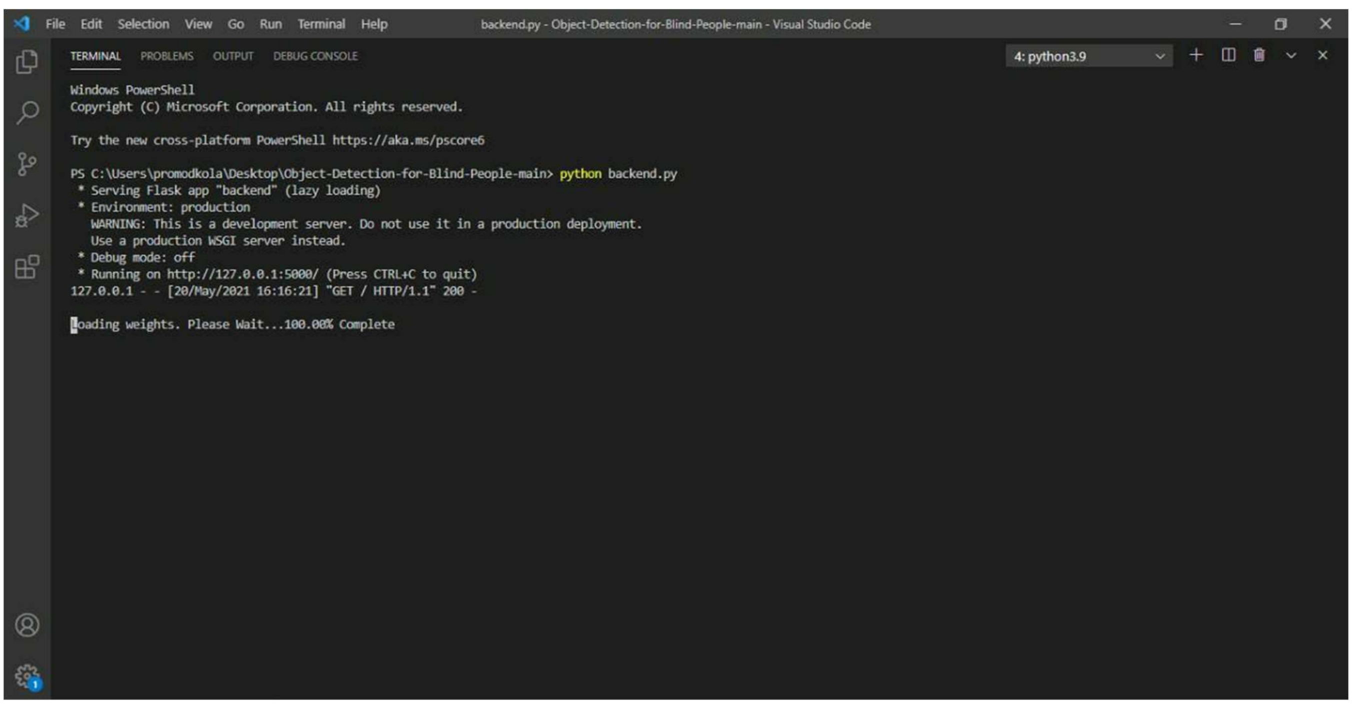

Figure 13: Training dataset

7.5 INPUT SCREENSHOTS

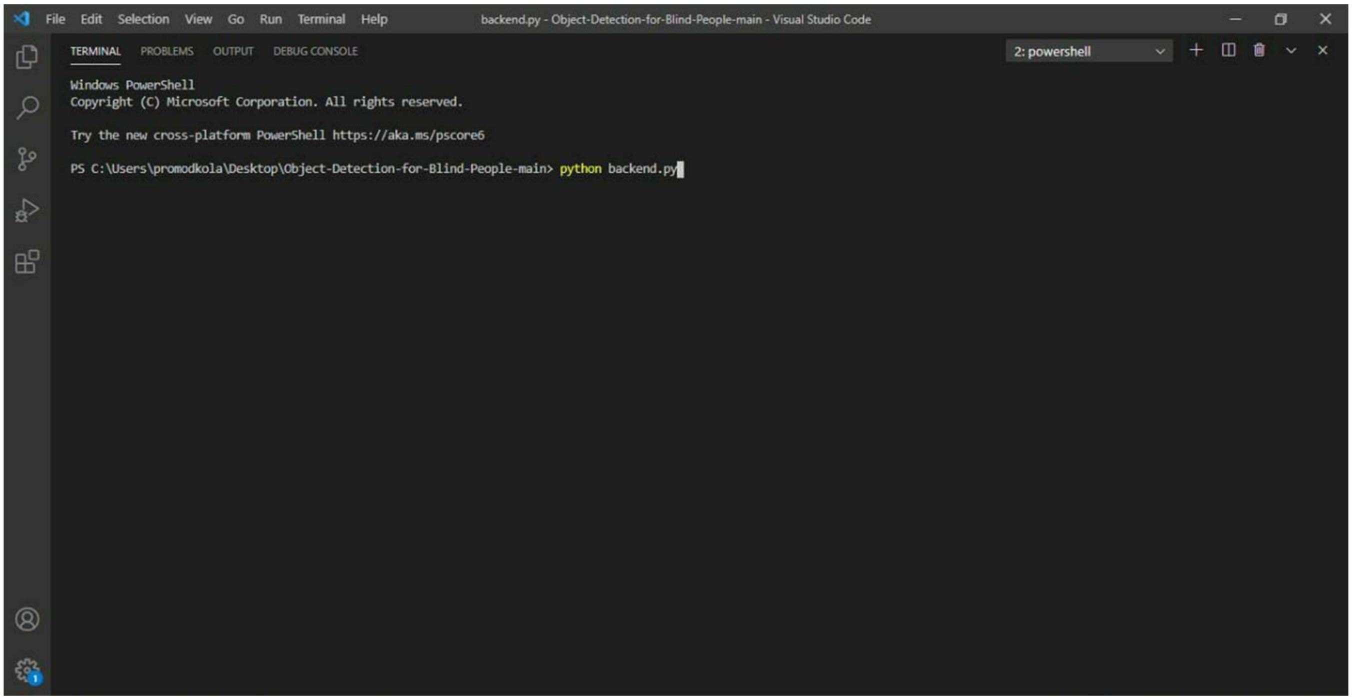

Figure 14: Command to run the project

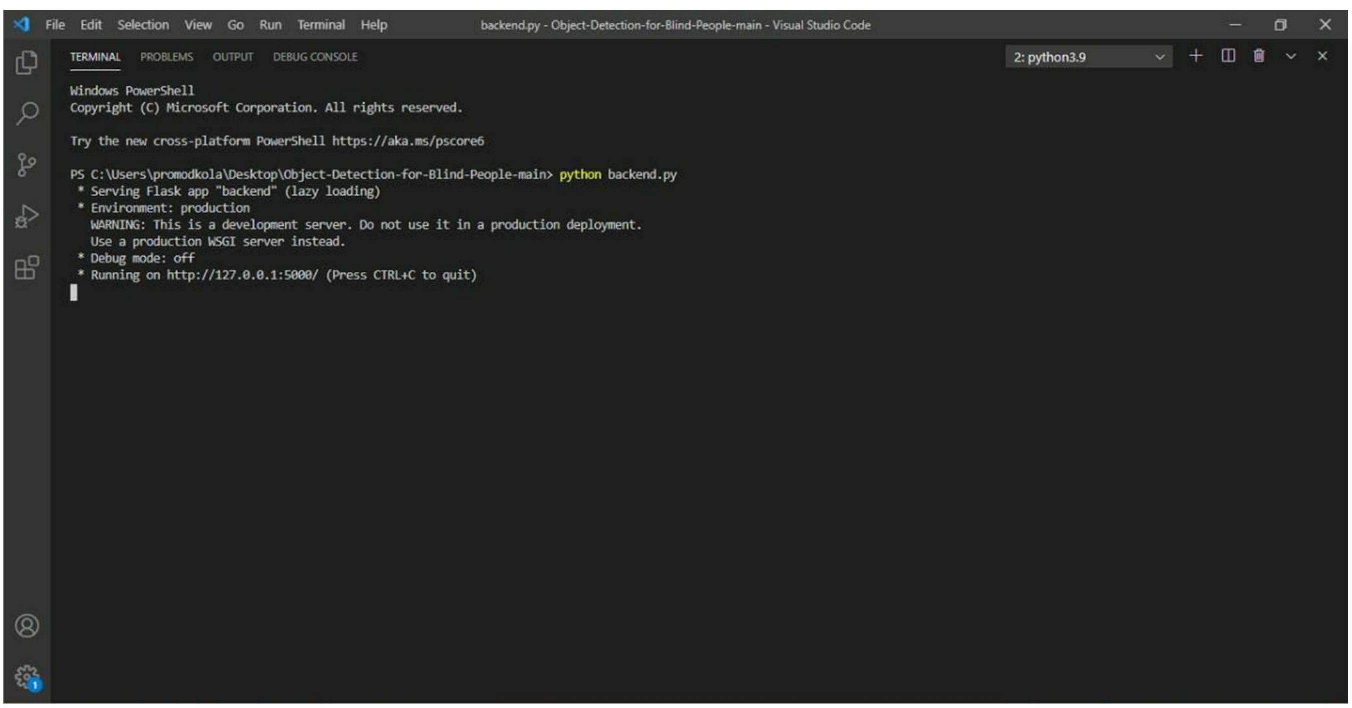

Figure 15: Website URL

7.6 OUTPUT SCREENSHOTS

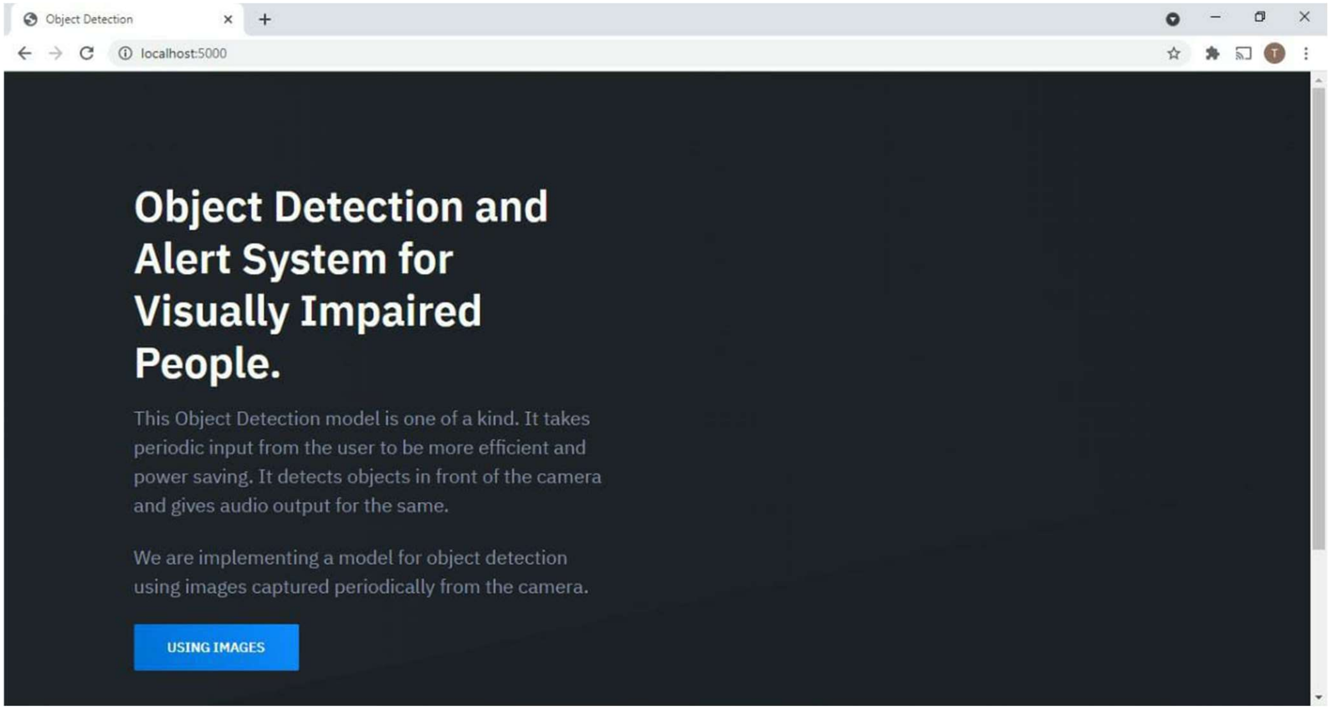

Figure 16: Website

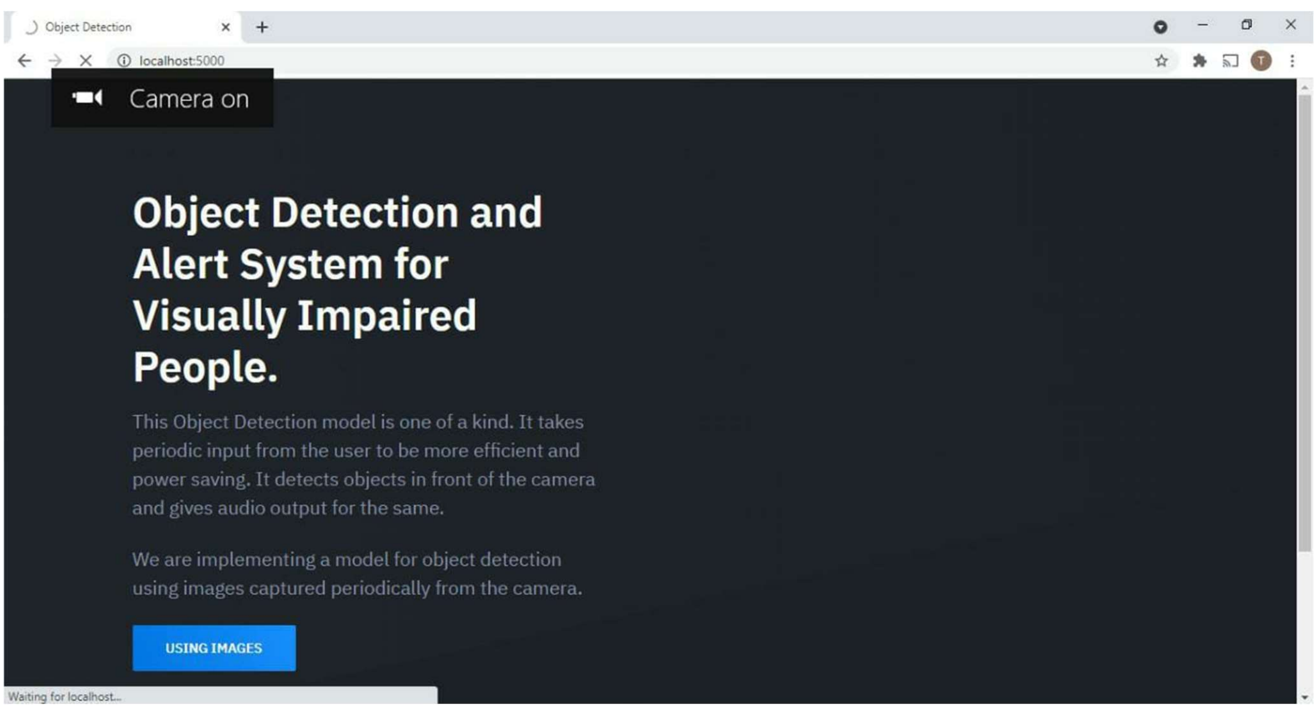

Figure 17: Image capturing

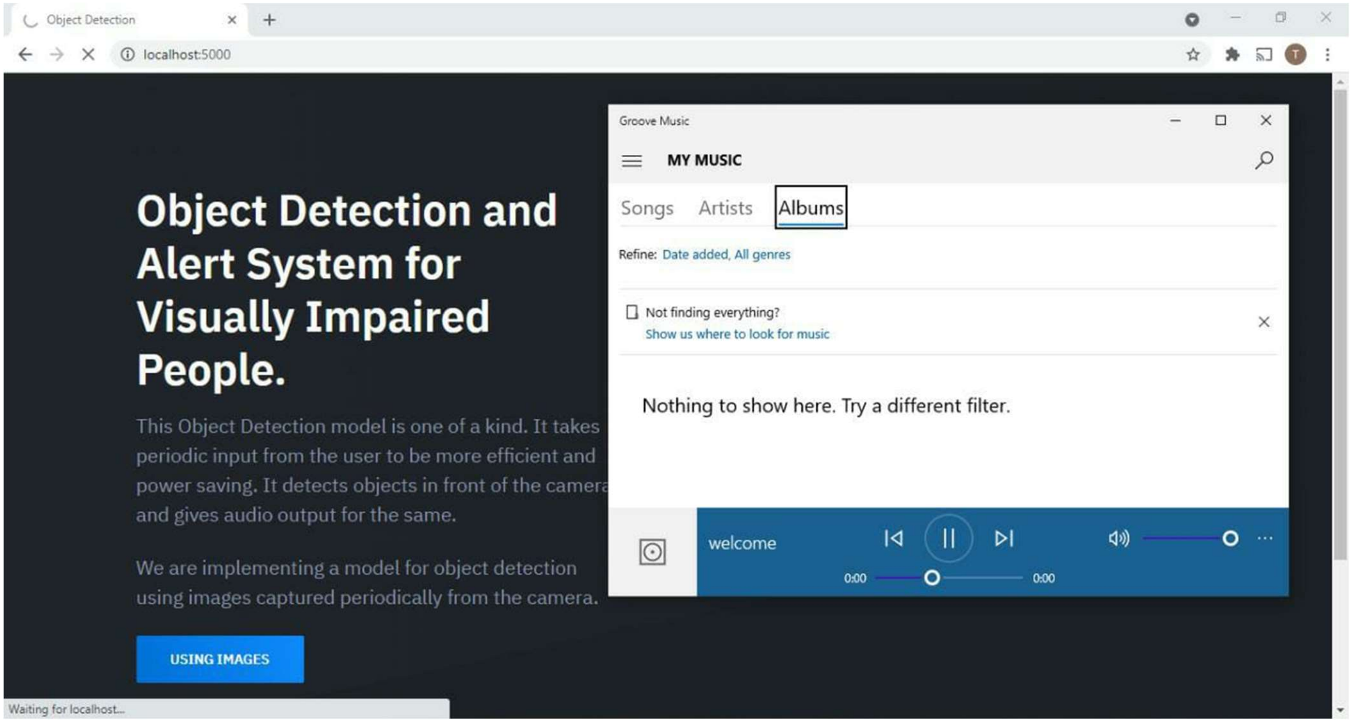

Figure 18: Object detection and audio output

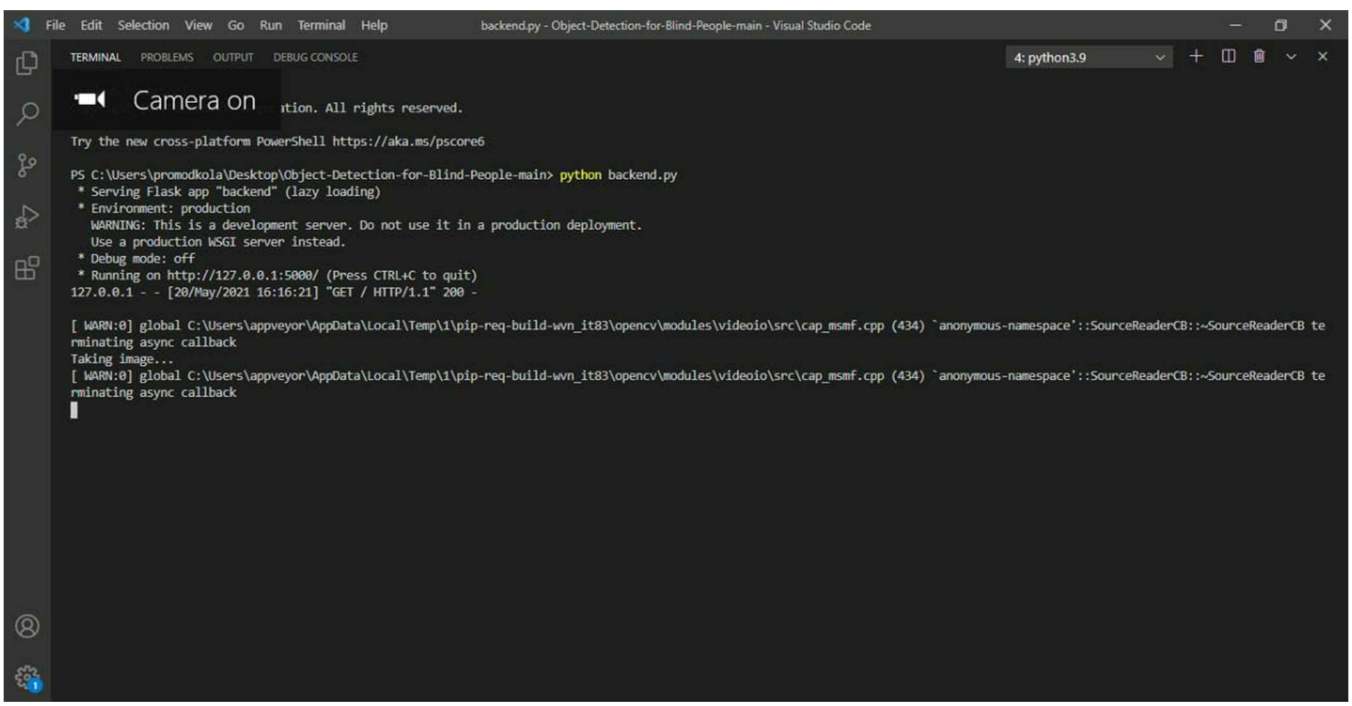

Figure 19: Image capturing in Visual Studio Code

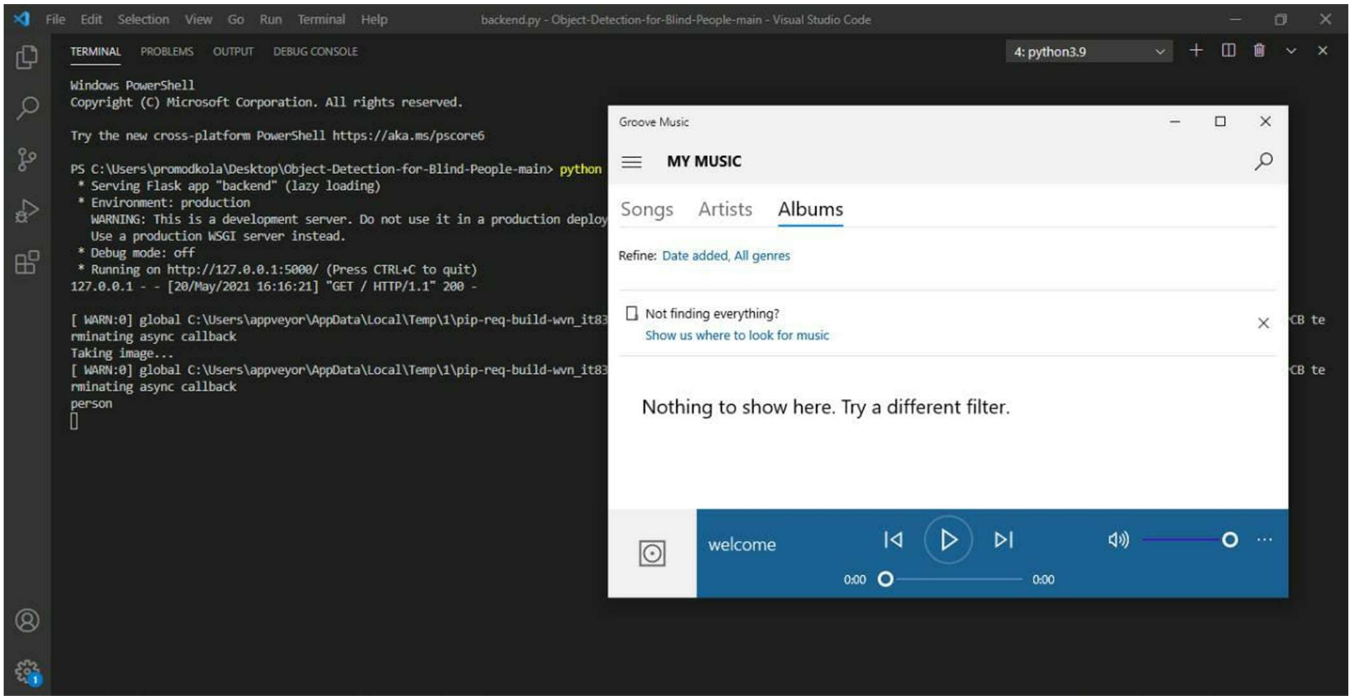

Figure 20: Object detection and audio output in Visual Studio Code

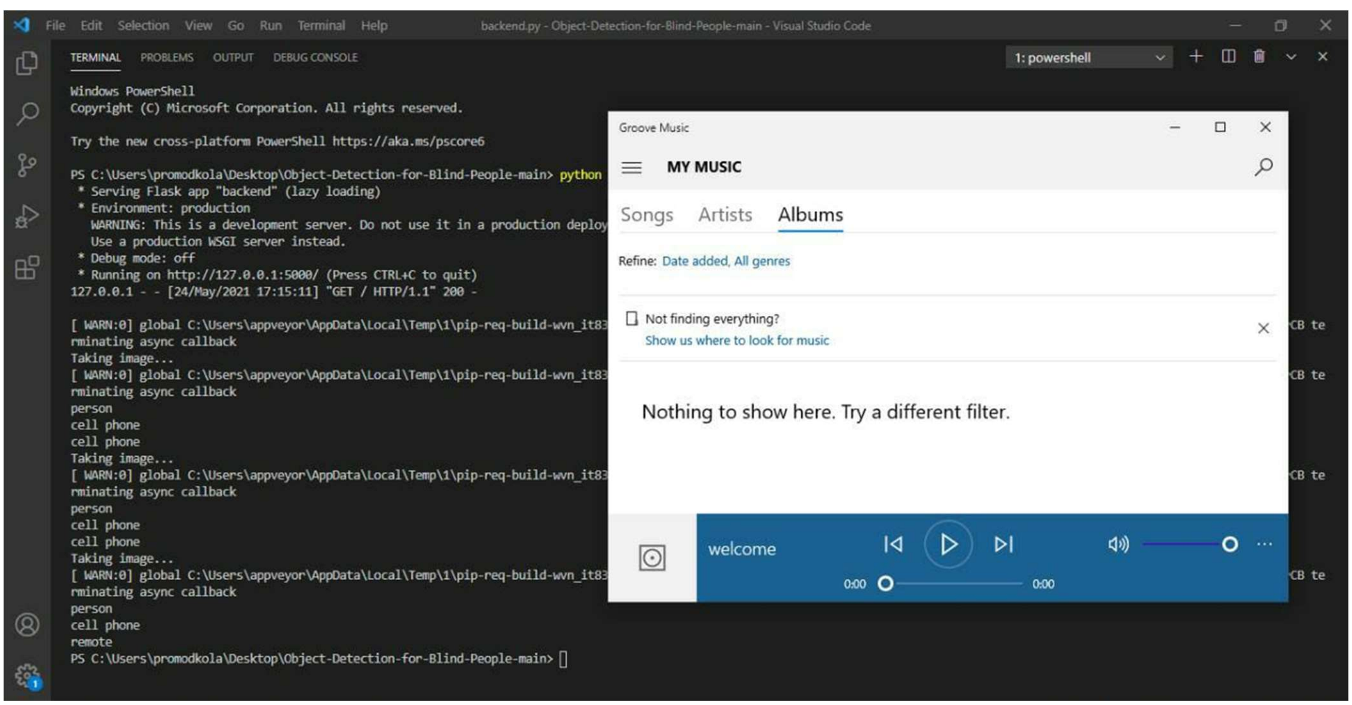

Figure 21: Detecting all the objects in the image captured

8. CONCLUSION AND FUTURE SCOPE

In this project, YOLOv3 has been applied and proposed to utilize for object identification in light of the fact that of its favourable circumstances. It was used to settle the genuine issue of navigation for the blind and visually impaired people in real time. An audio alert system was also included that will tell the user about the objects in front. This project is integrated into a web app for better usability. The visually impaired person will be able to sense and feel the environment in a better way using our app. This project can be further scaled and integrated with other accessories used by blind people such as their walking stick.

9. REFERENCES

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, “You Only Look Once: Unified, Real-Time Object Detection”,The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016

[2] B. Jason, A Gentle Introduction to Computer Vision, 2019. [Online]. Available: www.machinelearningmastery.com/what-is-computer-vision/

[3] Brian Katz, F.G. et al, 2012. NAVIG: Guidance system for the visually impaired using virtual augmented reality. In Technology and Disability

[4] Jafri, R., Ali, SA., Arabnia, HR., and Fatima, S., 2013. Computer vision-based object recognition for the visually impaired in an indoors environment: a survey. In The Visual Computer, Springer Berlin Heidelberg

[5] George Brandon Foshee, Timothy Allen Zigler, https://patents.google.com/patent/US20130044005

[6] Technology Dynamics Inc, https://patents.google.com/patent/US20130250078A1/en

[7] System and method for object detection, https://patents.google.com/patent/US9064172

[8] Digital object recognition audio-assistant for the visually impaired https://patents.google.com/patent/US20050208457A1/en

[9] Object Detection Device, https://portal.unifiedpatents.com/patents/patent/US-8803699-B2

[10] CVPR 2016 paper, https://openaccess.thecvf.com/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html