ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr. K. Madhuri**, **Head of the Department,** CSE(AIML), Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **Mrs. P. VAISHALI, Assistant Professor** for her technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

Pattern recognition (PR) is realized as a human recognition process which can be completed by computer technology. We should first enter useful information of identifying the object into the computer. For this reason, we must abstract the recognition object and establish its mathematical model to describe it and replace the recognition object for what the machine can process. The description of this object is the pattern. Simply speaking, the pattern recognition is to identify the category to which the object belongs, such as the face in face recognition, number plates in vehicles. This project out aim is to build an object detection model using YOLOv5.

Overall, YOLOv5 is a powerful and efficient object detection model that has achieved state-of-the-art results on several benchmarks, and is widely used in a variety of applications.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER – 1
INTRODUCTION

1.1 PROBLEM STATEMENT

In this project out aim is to build an object detection model using YOLOv5 (You Look Only Once). YOLO uses a single neural network to process the entire picture, then separates it into parts and predicts bounding boxes and probabilities for each component. These bounding boxes are weighted by the expected probability. The method "just looks once" at the image in the sense that it makes predictions after only one forward propagation run through the neural network. It then delivers detected items after non-max suppression (which ensures that the object detection algorithm only identifies each object once). Overall , YOLOv5 is a powerful and efficient object detection model that has achieved state-of-the-art results on a number of benchmarks, and is widely used in a
variety of applications.

1.2 MOTIVATION

Consider there is an unknown object in front and you do not know what the object corresponds to, object detection algorithms come into action at this time. It would be nice to know an unknown object within one click through object detection (computer vision), Google lens is a good example for this. Also, object detection is useful in many other areas where we can identify certain objects with help of a computer device or a mobile device. The key idea behind YOLOv5 is the use of a single neural network to simultaneously predict the bounding boxes and class probabilities for object in an image.

1.3 SCOPE

Object detection is breaking into a wide range of industries, including computer vision,

image retrieval, security, surveillance, automated vehicle systems, and machine inspection. It helps to identify an unknow image through computer vision.

Although the possibilities are endless when it comes to future use cases for object detection, there are still significant challenges remaining.

Herewith are some of the main useful applications of object detection: Vehicle's Plates recognition, self-driving cars, tracking objects, face recognition, medical imaging, object counting, object extraction from an image or video, person detection.

1.4 OUTLINE

The script first loads the dataset from a file, then it trains the model based on the input in the train split. It then provides an output with a confidence level based on its learning. The output basically gives an idea of what the object is present in the given image.

After that, the script splits the data into training and validation sets, and trains the model on the training data. Yolo uses a single neural network to process the entire picture, then separates it into parts and predicts bounding boxes and probabilities for each component. These bounding boxes are weighted by the expected probability.

CHAPTER – 2

LITERATURE SURVEY

EXISTING SYSTEM:

In the existing system, the detection of objects is slow compared to our system. YOLO v5 is different from all other prior releases, as this is a PyTorch implementation rather than a fork from original darknet. The major improvements include mosaic data augmentation and auto learning bounding box anchors. YOLO v5 is small. Generally, a weights file for YOLO v4 is 244 megabytes, but for YOLO v5 the file size is 27 megabytes.

PROPOSED SYSTEM:

Our unified architecture is extremely fast. Our base YOLO model processes images in real time at 45 frames per second. While still achieving the double the map of other real time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. However, based on the rapid pace of development in the field of object detection and classification, it is likely that YOLOv5 will introduce further improvements and innovations over the existing system. It is possible that YOLOv5 will incorporate new techniques and technologies. Finally, YOLO learns very general representations of objects. It out performs other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Overall Description:

This SRS is an overview of the whole project scenario. This document is to present a detailed description of the course management system. It will explain the purpose and features of the system, the interfaces of the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

3.2. Operating Environment:

Software Requirements:

Operating System	:	Windows 7 (Min)
Back End	:	Python

Hardware Requirements:

Processor	:	Intel CORE i5 10th gen
Speed	:	2.9 GHz (Min)
RAM	:	4 GB (Min)
Hard Disk	:	128 GB (Min)

3.3 Functional Requirements:

User Functionality:

- The user will be able to upload images to know the identification of the image.
- The user will be able to insert images up to 200 mb in size.

Admin Functionality:

- The admin manages the website.
- The admin can make changes to the website such as modifying the UI and making it moreinteractive than earlier.
- The admin can implement a better algorithm if at all a better algorithm is created in future.

3.4 Non-Functional Requirements:

3.4.1 Performance Requirements:

Performance requirements refer to static numerical requirements placed on theinteraction between the users and the software.

Response Time:

Average response time shall be less than 5 sec.

Recovery Time:

In case of system failure, the redundant system shall resume operations within30 secs. Average repair time shall be less than 45 minutes.

Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up.

Capacity:

The system accommodates 1 user at a time.

Utilization of Resources:

The system returns a result after a input is provided by the user.

3.4.2 Safety Requirements:

-NA-

3.4.2 Security Requirements:

The model will be running on a secure website i.e., an HTTPS website and also on a secure browser such as Google Chrome, Brave, etc.

3.4.3 Software Quality Attributes:

Reliability:

The system shall be reliable i.e., in case the webpage crashes, progress will be saved.

Availability:

The website will be available to all its users round the clock i.e., they can access the website at any time.

Security:

The model will be running on a secure website i.e., an HTTPS website and on a secure browser such as Google Chrome, Brave, etc.

Maintainability:

The model shall be designed in such a way that it will be very easy to maintain it in future. Our model is a neural network model and a web-based system and will depend much on the web server and on the neural networks. However, the web application will be designed using MobileNetV2 which is based on neural network approach and proper database modeling along with extensive documentation which will make it easy to develop, troubleshoot and maintain in future.

Usability:

The interfaces of the system will be user friendly enough that every user will be able to use it easily.

Scalability:

The system will be designed in such a way that it will be extendable. If more species or algorithms are going to be added in the system, then it would easily be done.

The same system can also be developed to become a mobile application rather than just a website.

CHAPTER-4

SYSTEM DESIGN

Use Case Diagram:

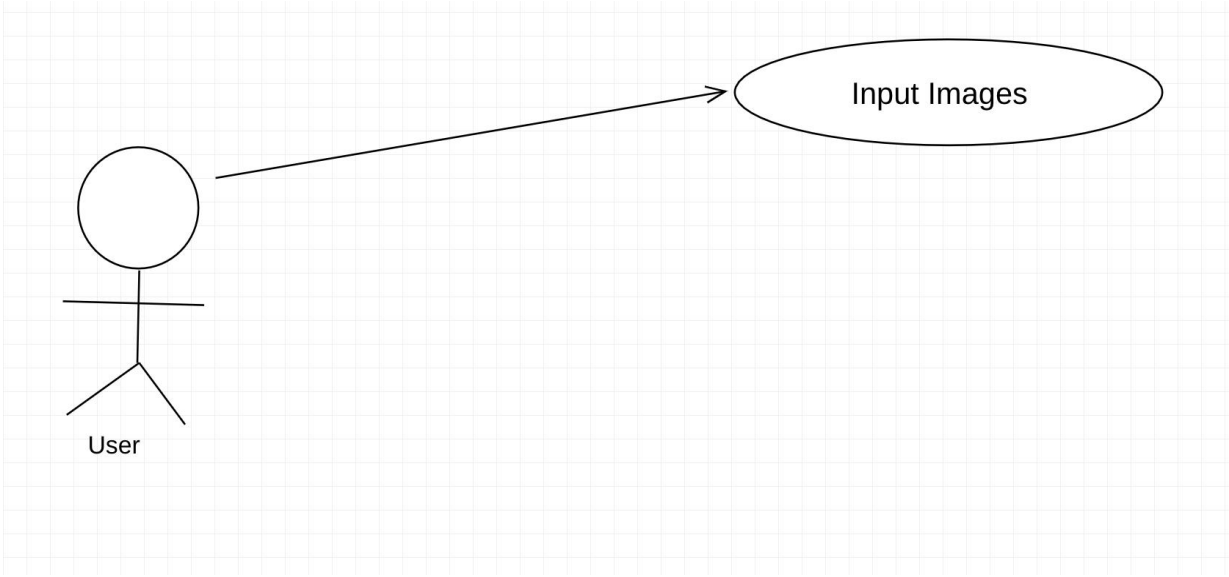

Fig.4.1.1 : Use Case diagram for User

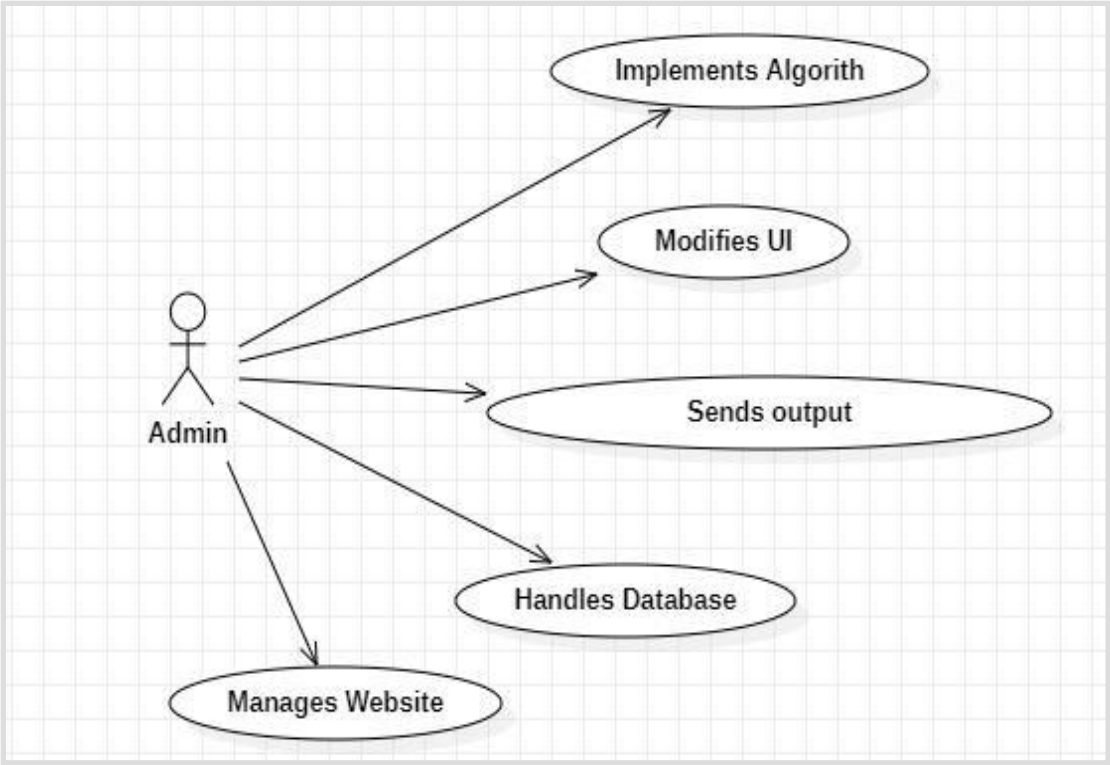

Fig.4.1.2 : Use Case diagram for User

Class Diagram:

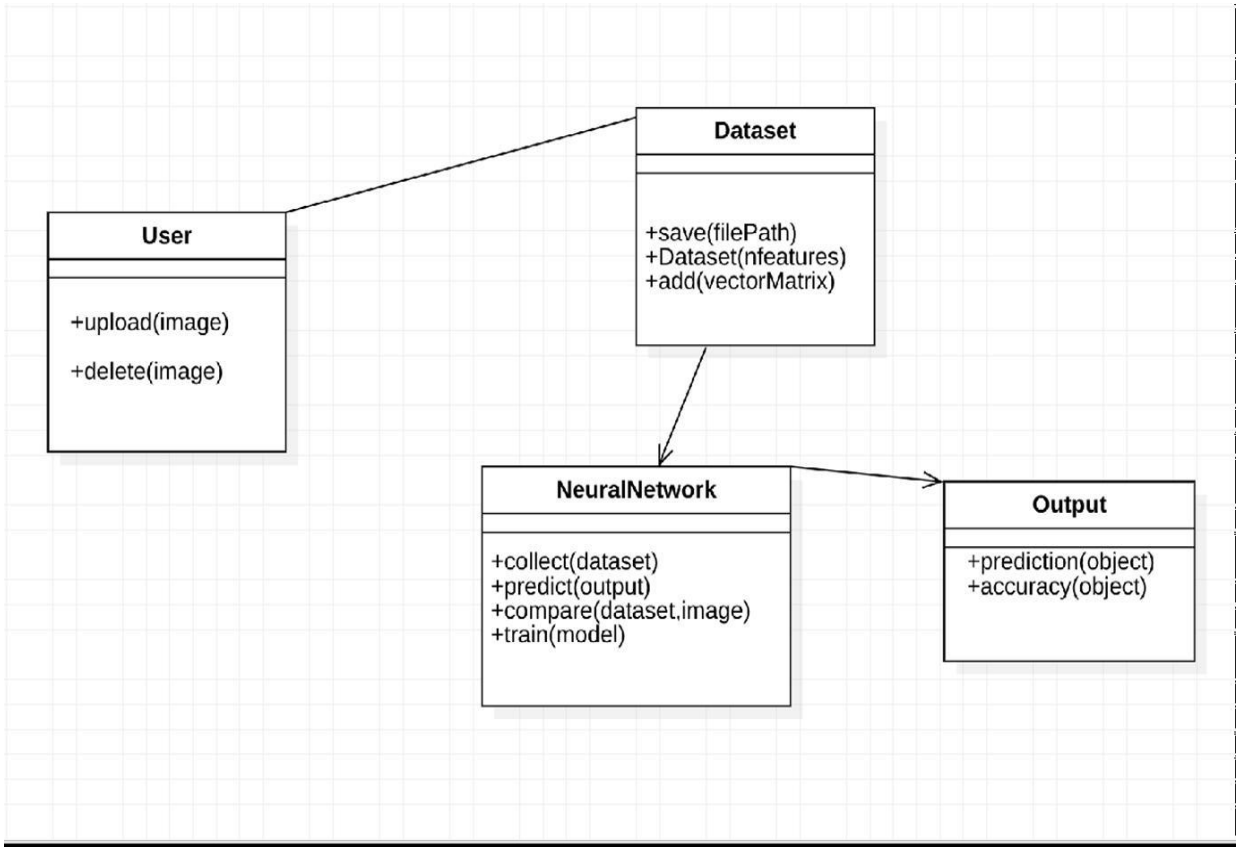

Fig.4.2 : Class diagram for Application

Sequence Diagram:

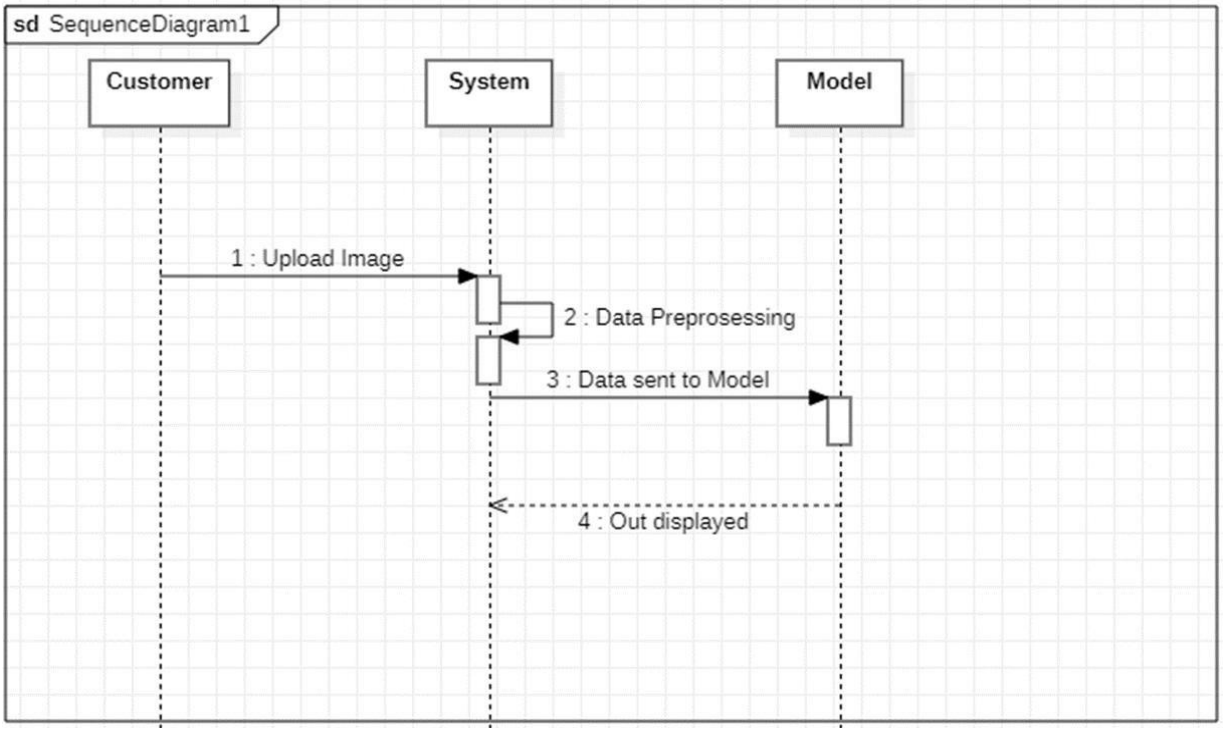

Fig4.3 : Sequence Diagram for Prediction.

Activity Diagram:

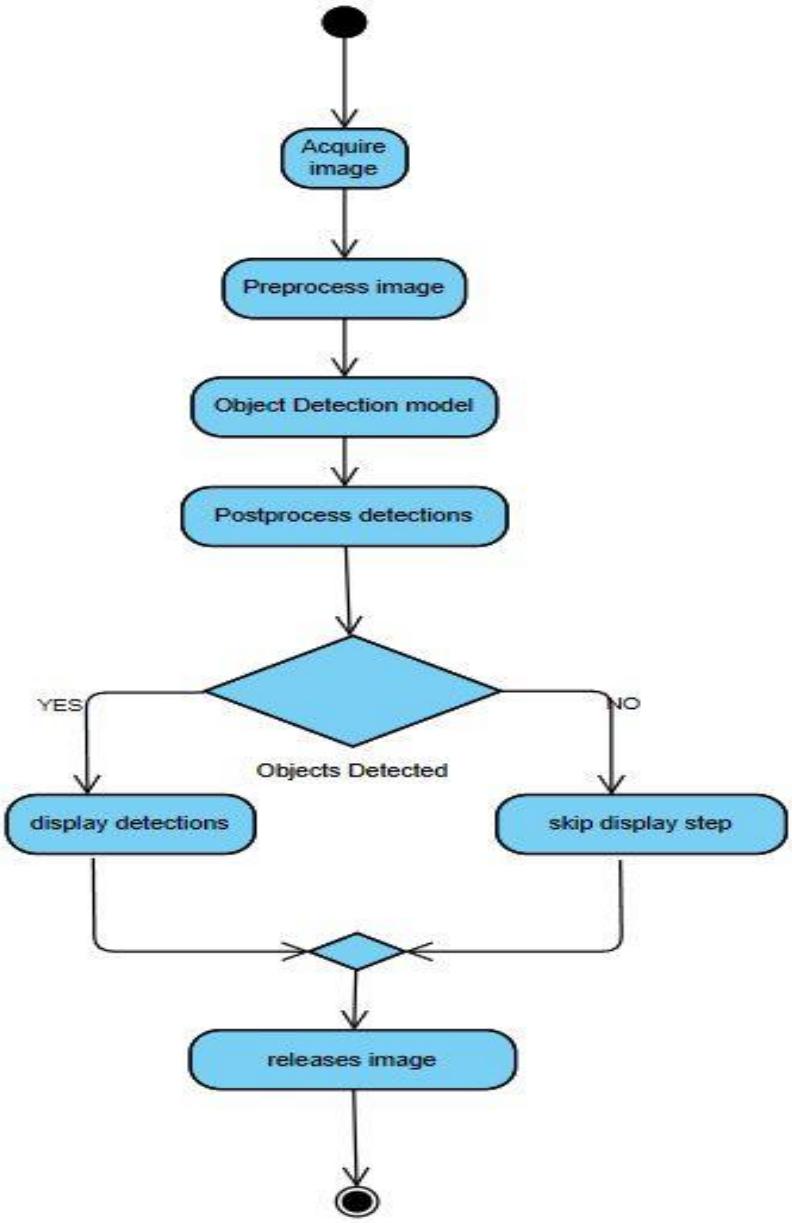

Fig.4.4 : Activity diagram for Workflow of model.

CHAPTER-5

IMPLEMENTATION

5.1 SAMPLE CODE

```
import os
import shutil
import time
from pathlib import Path

# import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random
from PIL import Image

from Apperaters_Detection_Main.apperaters_utils.utils import encodeImageIntoBase64

import sys
sys.path.insert(0, 'Apperaters_Detection_Main/predictor_yolo_detector')

from Apperaters_Detection_Main.predictor_yolo_detector.models.experimental import
attempt_load
from Apperaters_Detection_Main.predictor_yolo_detector.utils.datasets import LoadStreams,
LoadImages
from Apperaters_Detection_Main.predictor_yolo_detector.utils.general import (
    check_img_size, non_max_suppression, apply_classifier, scale_coords,
    xyxy2xywh, plot_one_box, strip_optimizer, set_logging)
```

```
from Apperaters_Detection_Main.predictor_yolo_detector.utils.torch_utils import
                select_device,
    time_synchronized



class Detector():
        # def _init_(self, weights, conf, source, img_size, save_dir,
                        save_txt, device, augment,agnostic_nms, conf_thres, ):

    def _init_(self, filename)

        path = os.getcwd()

        print(path)
        self.weights = "./Apperaters_Detection_Main/predictor_yolo_detector/best.pt"
        self.conf = float(0.5)
        self.source = "./Apperaters_Detection_Main/
                        predictor_yolo_detector/inference/images/"
        self.img_size = int(416)
        self.save_dir = "./Apperaters_Detection_Main/
                        predictor_yolo_detector/inference/output"
        self.view_img = False
        self.save_txt = False
        self.device = 'cpu'
        self.augment = True
        self.agnostic_nms = True
        self.conf_thres = float(0.5)
        self.iou_thres = float(0.45)
        self.classes = 0
        self.save_conf = True
        self.update = True
        self.filename = filename
```

```
 def detect(self, save_img=False):
 out, source, weights, view_img, save_txt, imgsz = \
     self.save_dir, self.source, self.weights, self.view_img, self.save_txt, self.img_size
 webcam = source.isnumeric() or source.startswith(('rtsp://', 'rtmp://', 'http://'))

 # Initialize
 set_logging()
 device = select_device(self.device)
 if os.path.exists(out):  # output dir
     shutil.rmtree(out)  # delete dir
os.makedirs(out)  # make new dir
half = device.type != 'cpu'  # half precision only supported on CUDA

# Load model
model = attempt_load(weights, map_location=device)  # load FP32 model
imgsz = check_img_size(imgsz, s=model.stride.max())  # check img_size
if half:
    model.half()  # to FP16

# Second-stage classifier
classify = False
if classify:
    modelc = load_classifier(name='resnet101', n=2)  # initialize
    modelc.load_state_dict(torch.load('weights/resnet101.pt', map_location=device)
   # load weights
    modelc.to(device).eval()

# Set Dataloader
vid_path, vid_writer = None, None
if webcam:
    view_img = True
```

```
    cudnn.benchmark = True  # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz)
else:
    save_img = True
    dataset = LoadImages(source, img_size=imgsz)

# Get names and colors
names = model.module.names if hasattr(model, 'module') else model.names
colors = [[random.randint(0, 255) for _ in range(3)] for _ in range(len(names))]



  # Run inference
        t0 = time.time()
        img = torch.zeros((1, 3, imgsz, imgsz), device=device)  # init img
        _ = model(img.half() if half else img) if device.type != 'cpu' else None
  # run once
        for path, img, im0s, vid_cap in dataset:
            img = torch.from_numpy(img).to(device)
            img = img.half() if half else img.float()  # uint8 to fp16/32
            img /= 255.0  # 0 - 255 to 0.0 - 1.0
            if img.ndimension() == 3:
                img = img.unsqueeze(0)

            # Inference
            t1 = time_synchronized()
            pred = model(img, augment=self.augment)[0]

            # Apply NMS
            pred = non_max_suppression(pred, self.conf)

            t2 = time_synchronized()
```

```
        # Apply Classifier
        if classify:
            pred = apply_classifier(pred, modelc, img, im0s)

        # Process detections
        for i, det in enumerate(pred):  # detections per image
            if webcam:  # batch_size >= 1
                p, s, im0 = path[i], '%g: ' % i, im0s[i].copy()
            else:
                p, s, im0 = path, '', im0s

save_path = str(Path(out) / Path(p).name)
txt_path = str(Path(out) / Path(p).stem) + ('_%g' % dataset.mode == 'video' )
s += '%gx%g ' % img.shape[2:]  # print string
gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
if det is not None and len(det):
    # Rescale boxes from img_size to im0 size
    det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()

    # Print results
    for c in det[:, -1].unique():
        n = (det[:, -1] == c).sum()  # detections per class
        s += '%g %ss, ' % (n, names[int(c)])  # add to string

    # Write results
    for *xyxy, conf, cls in reversed(det):
        if save_txt:  # Write to file
            xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
    # normalized
            line = (cls, conf, *xywh) if self.save_conf else (cls, *xywh)  # label format
            with open(txt_path + '.txt', 'a') as f:
                f.write(('%g ' * len(line) + '\n') % line)
```

```
        if save_img or view_img:  # Add bbox to image
            label = '%s %.2f' % (names[int(cls)], conf)
            plot_one_box(xyxy, im0, label=label, color=colors[int(cls)], line_thickness=3)

# Print time (inference + NMS)
print('%sDone. (%.3fs)' % (s, t2 - t1))

# Stream results
# if view_img:
#     cv2.imshow(p, im0)
#     if cv2.waitKey(1) == ord('q'):  # q to quit
#         raise StopIteration

 # Save results (image with detections)
 if save_img:
     if dataset.mode == 'images':
         im = Image.fromarray(im0)
         im.save("output.jpg")
         # cv2.imwrite(save_path, im0)
     else:
         print("Video Processing Needed")
             # if vid_path != save_path:  # new video
         #     vid_path = save_path
         #     if isinstance(vid_writer, cv2.VideoWriter):
         #         vid_writer.release()  # release previous video writer
         #     fourcc = 'mp4v'  # output video codec
         #     fps = vid_cap.get(cv2.CAP_PROP_FPS)
         #     w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
         #     h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
         #     vid_writer = cv2.VideoWriter(save_path, cv2.
              VideoWriter_fourcc(*fourcc), fps, (w, h))
```

```
            # vid_writer.write(im0)

            if save_txt or save_img:
                print('Results saved to %s' % Path(out))

            print('Done. (%.3fs)' % (time.time() - t0))

            return "Done"

        def detect_action(self):
            import os
        path = os.getcwd()
        print(path)
        with torch.no_grad():
            # if self.update:  # update all models (to fix SourceChangeWarning)
            #     for self.weights in ['yolov5s.pt', 'yolov5m.pt', 'yolov5l.pt', 'yolov5x.pt']:
            #         self.detect()
            #         strip_optimizer(self.weights)
            # else:
            self.detect()
        # /home/paul/PycharmProjects/factory_fire_&_smoke/your_file.jpg
        #imagekeeper = []
        opencodedbase64 = encodeImageIntoBase64("output.jpg")
        result = {"image": opencodedbase64.decode('utf-8')}
        return result

    # firensmoke = Detector("inputimage.jpg")
    # firensmoke.detect_action()

def bbox_iou(box1, box2, x1y1x2y2=True, GIoU=False, DIoU=False,
                     CIoU=False, eps=1e-9):
    # Returns the IoU of box1 to box2. box1 is 4, box2 is nx4
```

```
box2 = box2.T

# Get the coordinates of bounding boxes
if x1y1x2y2:  # x1, y1, x2, y2 = box1
    b1_x1, b1_y1, b1_x2, b1_y2 = box1[0], box1[1], box1[2], box1[3]
    b2_x1, b2_y1, b2_x2, b2_y2 = box2[0], box2[1], box2[2], box2[3]
else:  # transform from xywh to xyxy
    b1_x1, b1_x2 = box1[0] - box1[2] / 2, box1[0] + box1[2] / 2
    b1_y1, b1_y2 = box1[1] - box1[3] / 2, box1[1] + box1[3] / 2
    b2_x1, b2_x2 = box2[0] - box2[2] / 2, box2[0] + box2[2] / 2
    b2_y1, b2_y2 = box2[1] - box2[3] / 2, box2[1] + box2[3] / 2

    # Intersection area
    inter = (torch.min(b1_x2, b2_x2) - torch.max(b1_x1, b2_x1)).clamp(0) * \
            (torch.min(b1_y2, b2_y2) - torch.max(b1_y1, b2_y1)).clamp(0)

    # Union Area
    w1, h1 = b1_x2 - b1_x1, b1_y2 - b1_y1 + eps
    w2, h2 = b2_x2 - b2_x1, b2_y2 - b2_y1 + eps
    union = w1 * h1 + w2 * h2 - inter + eps

    iou = inter / union
    if GIoU or DIoU or CIoU:
        cw = torch.max(b1_x2, b2_x2) - torch.min(b1_x1, b2_x1
        ch = torch.max(b1_y2, b2_y2) - torch.min(b1_y1, b2_y1)  # convex height
        if CIoU or DIoU:  # Distance or Complete IoU
            c2 = cw ** 2 + ch ** 2 + eps  # convex diagonal squared
            rho2 = ((b2_x1 + b2_x2 - b1_x1 - b1_x2) ** 2 +
                    (b2_y1 + b2_y2 - b1_y1 - b1_y2) ** 2) / 4  # center distance squared
            if DIoU:
                return iou - rho2 / c2  # DIoU
            elif CIoU:  # https://github.com/Zzh-tju/DIoU-SSD-
```

```
                v = (4 / math.pi ** 2) * torch.pow(torch.atan(w2 / h2) /
                with torch.no_grad():
                    alpha = v / ((1 + eps) - iou + v)
                return iou - (rho2 / c2 + v * alpha)  # CIoU
            else:  # GIoU https://arxiv.org/pdf/1902.09630.pdf
                c_area = cw * ch + eps  # convex area
                return iou - (c_area - union) / c_area  # GIoU
        else:
            return iou



def plot_one_box(x, img, color=None, label=None, line_thickness=None):
    # Plots one bounding box on image img
    tl = line_thickness or round(0.002 * (img.shape[0] + img.shape[1]) / 2) + 1  #
    line/font thickness
    color = color or [random.randint(0, 255) for _ in range(3)]
    c1, c2 = (int(x[0]), int(x[1])), (int(x[2]), int(x[3]))
    cv2.rectangle(img, c1, c2, color, thickness=tl, lineType=cv2.LINE_AA)
    if label:
        tf = max(tl - 1, 1)  # font thickness
        t_size = cv2.getTextSize(label, 0, fontScale=tl / 3, thickness=tf)[0]
        c2 = c1[0] + t_size[0], c1[1] - t_size[1] - 3
        cv2.rectangle(img, c1, c2, color, -1, cv2.LINE_AA)  # filled
        cv2.putText(img, label, (c1[0], c1[1] - 2), 0, tl / 3, [225, 255, 255], thickness=tf,
    lineType=cv2.LINE_AA)


    #Flask
    from flask import Flask, request, jsonify, render_template, Response
    import os
    from flask_cors import CORS, cross_origin
    from Apperaters_Detection_Main.apperaters_utils.utils import decodeImage
    from Apperaters_Detection_Main.predictor_yolo_detector.detector_test import
```

```
    Detector

# import sys
# sys.path.insert(0, './Apperaters_Detection_Main')

os.putenv('LANG', 'en_US.UTF-8')
os.putenv('LC_ALL', 'en_US.UTF-8')

app = Flask(_name_)
CORS(app)


# @cross_origin()
class ClientApp:
    def _init_(self):
        self.filename = "inputImage.jpg"
        # modelPath = 'research/ssd_mobilenet_v1_coco_2017_11_17'
        self.objectDetection = Detector(self.filename)


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


@app.route("/predict", methods=['POST', 'GET'])
@cross_origin()
def predictRoute():
    try:
        image = request.json['image']
        decodeImage(image, clApp.filename)
        result = clApp.objectDetection.detect_action()
```

```
    except ValueError as val:
        print(val)
        return Response("Value not found inside  json data")
    except KeyError:
        return Response("Key value error incorrect key passed")
    except Exception as e:
        print(e)
        result = "Invalid input"

    return jsonify(result)



# port = int(os.getenv("PORT"))
if _name_ == "_main_":
    clApp = ClientApp()
    port = 9500
    app.run(host='0.0.0.0', port=port)

#Html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Warehouse Apparel</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
     integrity="sha384-
    Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/d
    AiS6JXm" crossorigin="anonymous">
```

```
<style>

 .iupload h3 {
        color: #1b2d6b;
        font-size: 20px;
        font-weight: 700;
 }

 .img-part-1 {
        height: 300px;
        width: 300px;
        margin: 0px auto;
 }

 .image-part {
        height: 300px;
        width: 300px;
        border: 1px solid #1b2d6b;
 }

 .image-part img {
        /* position: absolute; */
        height: 300px;
        width: 300px;
        display: none;
        padding: 5px;
 }

 .image-part #video {
        /* display: block; */
        height: 300px;
        width: 300px;
```

```
        padding: 5px;
}

.res-part {
        /* margin-left: 20px; */
        height: 400px;
        width: 100%;
        padding: 5px;
        margin: 0px auto;
        overflow: auto;
}

.upload-image {
        /* margin-left: 20px; */
        height: 400px;
        width: auto;;
        padding: 5px;
        margin: 0px auto;
        overflow: auto;
}

.resp-img {
        height: 400px;
        width: auto;
        margin: 0px auto;
}

.jsonRes {
        margin-left: 30px;
}

#send {
```

```
        cursor: pointer;
}

.btn-part {
        width: 325px;
}

textarea,
select,
.form-control,
.custom-select,
button.btn,
.btn-primary,
input[type="text"],
input[type="url"],
.uneditable-input {
        border: 1px solid #363e75;
        outline: 0 !important;
        border-radius: 0px;
        box-shadow: none;
        -webkit-box-shadow: none;
        -moz-box-shadow: none;
        -moz-transition: none;
        -webkit-transition: none;
}

textarea:focus,
select:focus,
.form-control:focus,
.btn:focus,
.btn-primary:focus,
.custom-select:focus,
```

```
input[type="text"]:focus,
.uneditable-input:focus {
        border: 1px solid #007bff;
        outline: 0 !important;
        border-radius: 0px;
        box-shadow: none;
        -webkit-box-shadow: none;
        -moz-box-shadow: none;
        -moz-transition: none;
        -webkit-transition: none;
}

#loading {
        position: fixed;
        left: 0px;
        top: 0px;
        width: 100%;
        height: 100%;
        z-index: 9999999999;
        overflow: hidden;
        background: rgba(255, 255, 255, 0.7);
}

.loader {
        border: 8px solid #f3f3f3;
        border-top: 8px solid #363e75;
        border-radius: 50%;
        width: 60px;
        height: 60px;
        left: 50%;
        margin-left: -4em;
        display: block;
```

```
            animation: spin 2s linear infinite;
        }

        .loader,
        .loader:after {
            display: block;
            position: absolute;
            top: 50%;
            margin-top: -4.05em;
        }

        @keyframes spin {
            0% {
                transform: rotate(0deg);
            }

            100% {
                transform: rotate(360deg);
            }
        }

        .logo {
            position: absolute;
            right: 0px;
            bottom: 0px;
            margin-right: 30px;
            margin-bottom: 30px;
        }
    </style>
</head>

<body>
```

```
<!-- <div class="main container">
        <section class="iupload">
                <h3 class="text-center py-4">Object Detection Using TFOD</h3>
                <div class="row">
                        <div class="img-part col-md-6">
                                <div class="image-part">
                                        <video autoplay id="video"
                                                poster="https://img.freepik.com/free-
vector/group-young-people-posing-photo_52683-
18824.jpg?size=338&ext=jpg"></video>
                                        <img src="" id="photo">
                                        <canvas style="display:none;"
id="canvas"></canvas>
                                </div>
                        </div>
                        <div class="col-md-6 col-xs-12 right-part">
                                <h5 class="mb-2">
                                        Prediction Results
                                </h5>
                                <div class="row">
                                        <div class="res-part2 col-md-2 col-xs-
12"></div>
                                </div>
                        </div>
                </div>
        </section>


</div> -->

<!-- Header -->
<header class="bg-primary text-center py-5 mb-4">
```

```
  <div class="container">
    <h1 class="font-weight-light text-white">Warehouse Apparel Detection using
  YOLOv5</h1>
  </div>
</header>

<!-- Page Content -->
<div class="container">


  <form class="input-group upload-data row">
        <div class="col-xl-6 col-md-6 col-sm-6">
        <button type="button" class="btn btn-primary col-12"
  id="uload">Upload</button>
        </div>
        <div class="col-xl-6 col-md-6 col-sm-6">
                <button id="send" type="button" class="btn btn-success col-
  12">Predict</button>
        </div>

        <!-- change url value  -->

        <input type="hidden" class="form-control mr-2" id="url" placeholder="Enter
  REST Api url..." value="../predict" />
        <input name="upload" type="file" id="fileinput" style="position:absolute;top:-
  500px; display: none;" /><br />
  </form>

  <div class="row">
    <!-- Team Member 1 -->
    <div class="col-xl-6 col-md-6 col-sm-6 mb-6">
        <div class="card border-0 shadow upload-image ">
```

```
          <!-- <img src="https://source.unsplash.com/TMgQMXoglsM/500x350"
  class="card-img-top" alt="..."> -->
                    <video autoplay id="video" poster="https://img.freepik.com/free-
  vector/group-young-people-posing-photo_52683-
  18824.jpg?size=338&ext=jpg"></video>
                    <img src="" class="" id="photo">
                    <canvas style="display:none;" id="canvas"></canvas>
          <!-- <div class="card-body text-center">
                    <h5 class="card-title mb-0">Team Member</h5>
          </div> -->
        </div>
    </div>
    <!-- Team Member 2 -->
    <div class="col-xl-6 col-md-6 col-sm-6 mb-6">
        <div class="card border-0 shadow res-part2">
          <div class="card-body text-center">
                    <h5 class="card-title mb-0">Prediction Results</h5>
          </div>
        </div>
    </div>
  </div>
  <!-- /.row -->


</div>
<!-- /.container -->



  <div id="loading">
        <div class="loader"></div>
```

```
    </div>
    <script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
            integrity="sha384-
    ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4
    Q" crossorigin="anonymous">
    </script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
            integrity="sha384-
    JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCm
    Yl" crossorigin="anonymous">
    </script>

</html>

<script>
            var mybtn = document.getElementById('startbtn');
            var myvideo = document.getElementById('video');
            var mycanvas = document.getElementById('canvas');
            var myphoto = document.getElementById('photo');
            var base_data = "";

            function sendRequest(base64Data) {
                        var type = "json";
                        if (base64Data != "" || base64Data != null) {
                                    if (type == "imgtobase") {
                                                $(".res-part").html("");
                                                $(".res-part").html(base64Data);
                                    } else if (type == "basetoimg") {
                                                var imageData = $("#imgstring").val();
```

```
                $(".res-part").html("");
                $(".res-part").append("<img
src='" + imageData + "' alt=" />");
            } else {
                var url = $("#url").val();
                $("#loading").show();
                $.ajax({
                    url: url,
                    type: "post",
                    cache: false,
                    async: true,
                    crossDomain: true,
                    headers: {
                        'Content-Type': 'application/json',
                        'Access-Control-Allow-Origin': '*'
                    },
                    data: JSON.stringify({
                        image: base64Data
                    }),
                    success: function (res) {
                        $(".res-part").html("");
                        $(".res-part2").html("");
                        var imageData = res.image;
                        $(".res-part2").append("<img class='resp-
img' src='" +
                            imageData + "' alt=" />");
                        // $(".res-part").html("<pre>" +
JSON.stringify(res[0], undefined, 2) + "</pre>");
                        $("#loading").hide();
                    }
                });
            }
```

```
        }
    }

    $(document).ready(function () {
        $("#loading").hide();

        $('#send').click(function (evt) {
            sendRequest(base_data);
        });

        $('#uload').click(function (evt) {
            $('#fileinput').focus().trigger('click');
        });
        $("#fileinput").change(function () {
            if (this.files && this.files[0]) {
                var reader = new FileReader();
                reader.onload = function (e) {
                    var url = e.target.result;
                    var img = new Image();
                    img.crossOrigin = 'Anonymous';
                    img.onload = function () {
                        var canvas =
document.createElement('CANVAS');
                        var ctx = canvas.getContext('2d');
                        canvas.height = this.height;
                        canvas.width = this.width;
                        ctx.drawImage(this, 0, 0);
                        base_data =
canvas.toDataURL('image/jpeg', 1.0).replace(
                            /^data:image.+;base64,/, '');
                        canvas = null;
                    };
```

```
                    img.src = url;
                    $('#photo').attr('src', url);
                    $('#photo').show();
                    $('#video').hide();
                }
                reader.readAsDataURL(this.files[0]);
            }
        });
    });
    </script>
</body>
```

CHAPTER – 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed, then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 Python Programs Integration Testing

Test Case to Collect Dataset and Load the Dataset

Test Case ID:	3
Test Case Name:	Collect Dataset and Load the Dataset
Purpose:	Check Dataset is collected, and the data is stored
Input:	Provide Dataset as input
Expected Result:	Dataset is collected and view the Dataset and store the Dataset
Actual Result:	Load the Dataset and view the Dataset and store
Failure	If the dataset is not loaded, it will throw an error.

Table 6.1.3 Collect Dataset and Load the Dataset

Test Case to check working of the commands provided by the user

Test Case ID:	4
Test Case Name:	Proper execution of the commands.
Purpose:	To check whether the system performs desired action for the commands provided.
Input:	Provide sonar frequencies.
Expected Result:	The system performs the action specified in the command.
Actual Result:	The system performs the action specified in the command.
Failure	If the system doesn't understand the command it will not perform the action.

Table 6.1.4 Execution of commands

CHAPTER - 7
SCREENSHOTS

Figure 7.1: Custom image 1(Results)S

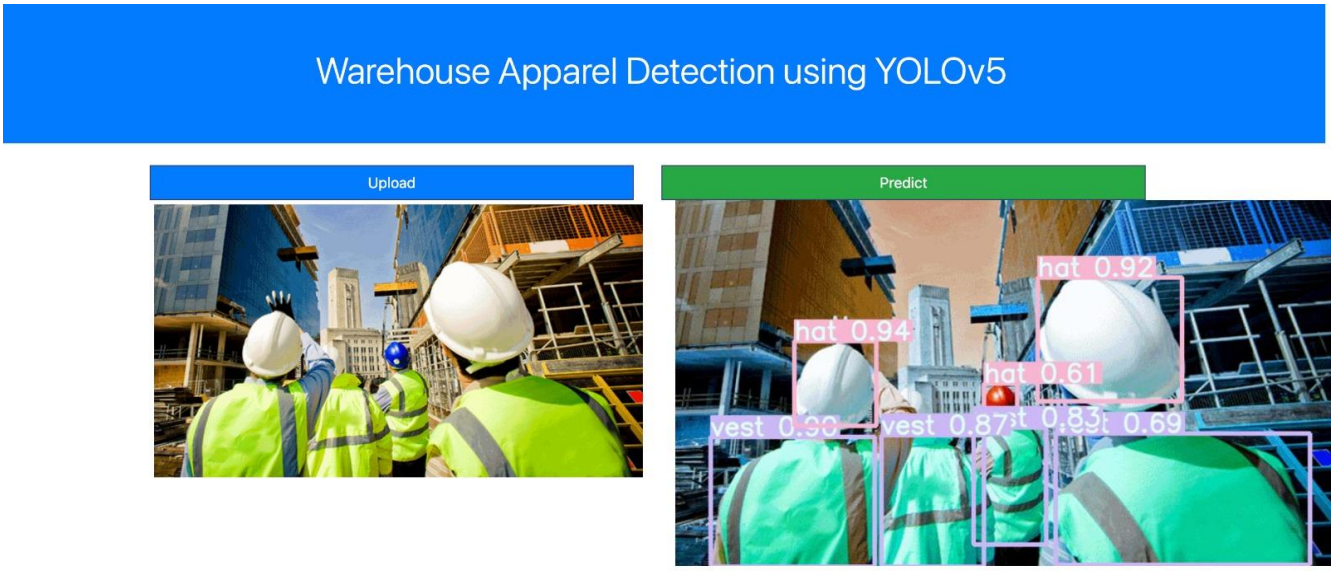

Figure7.2: Custom image 2 (Results)

CHAPTER - 8

CONCLUSION AND FUTURE SCOPE

- Object detection is inextricably linked to other similar computer vision techniques like image recognition and image segmentation, in that it helps us understand and analyze scenes in images or video. But there are important differences. Image recognition only outputs a class label for an identified object, and image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability to locate objects within an image or video. This then allows us to count and then track those objects. Given these key distinctions and object detection's unique capabilities, we can see how it can be applied in a number of ways:
- Autonomous vehicles: YOLOv5 can be used to detect pedestrians, vehicles, traffic signs, and other objects in real-time video feeds from cameras mounted on self-driving cars.

- Security and surveillance: YOLOv5 can be used to detect people, vehicles, and other objects in security camera footage, and to alert authorities or take other actions in response to suspicious activity.

- Robotics: YOLOv5 can be used to enable robots to "see" and understand their environment, allowing them to navigate, avoid obstacles, and interact with objects and people.

- Augmented reality: YOLOv5 can be used to detect and track objects in real-time video feeds, enabling the overlay of digital information or graphics on top of the physical world.

- Of course, this is not an exhaustive list, but it includes some of the primary ways in which object detection is shaping our future. Overall, the scope of object detection using YOLOv5 is vast and limited only by the creativity and imagination of the developers using it.

BIBLIOGRAPHY

1. *Roberts, Soraya (December 16, 2011). "Zac Efron Adopts Drake's 'YOLO' Motto, as Does Souljaboy". Yahoo! OMG! CA. Retrieved November 8, 2012.*
2. ^ *"You Only Live Once – YOLO". Quote Investigator. May 24, 2012. Retrieved August 4, 2012.*
3. ^ *CRITIC, JOEL SELVIN, CHRONICLE POP MUSIC (1996-05-26). "Mickey Hart Marches on to His Own Beat / Ex-Grateful Dead drummer about to release magnum opus". SFGATE. Retrieved 2021-02-23.*
4. ^ *Bereznak, Alyssa. "An Oral History of YOLO, the Word That Lived Too Long". Vanity Fair. Retrieved 2021-02-23.*
5. ^ *"Skins | Ep. 107 | Songs from the Show". MTV. August 14, 2014. Retrieved August 25, 2014.*
6. ^ *Swanson, Mirjam (May 18, 2012). "Track and Field: Mitchells making most of it". Press-Enterprise. Retrieved October 10, 2012.*
7. ^ *Dye, Kevin. "Class of 2012 calls it a year". The Madison Press. Archived from the original on June 8, 2012. Retrieved March 3, 2013.*

.

APPENDIX A: TOOLS AND TECHNOLOGIES

- PYTHON V3: The Python language comes with many libraries and frameworks that make coding easy. This also saves a significant amount of time.
- YOLOv5: It is a family of compound-scaled object detection models trained on the COCO dataset, and includes simple functionality for Test Time Augmentation (TTA), model ensembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite.
- JUPYTER NOTEBOOK: The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.
- NUMPY: NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- MATPLOTLIB: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- WINDOWS 11 : Windows 11 was used as the operating system.