"OBJECT DETECTION FROM CCTV"

Major Project Report

Submitted in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL ENGINEERING

By

Vinamra Gupta

(20BEE129)

Department of Electrical Engineering
Institute of Technology
NIRMA UNIVERSITY
Ahmedabad 382 481

May 2024

CERTIFICATE

This is to certify that the Minor Project Report entitled "Object Detection From CCTV" submitted by Mr. Vinamra Gupta (20BEE129) towards the partial fulfillment of the requirements for the award of a degree in Bachelor of Technology in the field of Electrical Engineering of Nirma University is the record of work carried out by him under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this major project work to the best of our knowledge have not been submitted to any other University or Institution for the award of any degree or diploma.

Date:

Institute – Guide

Prof. Samyak Shah
Department of Electrical Engineering
Institute of Technology
Nirma University
Ahmedabad

Head of Department
Department of Electrical Engineering
Institute of Technology
Nirma University
Ahmedabad

Director
Institute of Technology
Nirma University
Ahmedabad

ACKNOWLEDGEMENT

I must acknowledge the strength, energy, and patience that almighty **GOD** bestowed upon me to start & accomplish this work with the support of all concerned, a few of whom I am trying to name hereunder.

I want to express my deep gratitude to *all faculties of the Electrical Engineering Department* for their valuable guidance and motivation.

I would like to express my sincere respect and profound gratitude to **Prof. (Dr.) S. C. Vora**, *Professor & Head of Department (Electrical Engineering)* for supporting and providing the opportunity for the summer internship.

I would also like to thank all my friends who have helped me directly or indirectly with the completion of my summer internship.

No words are adequate to express my indebtedness to my parents and for their blessings and good wishes. To them, I bow in the deepest reverence.

-Vinamra Gupta(20BEE129)

ABSTRACT

This project focuses on the development of an intelligent object detection system designed for CCTV footage as well as the detection of the speed of vehicles. Its goal is to enhance surveillance capabilities by integrating advanced computer vision, deep learning, and data analysis techniques. The project initiates an in-depth examination of existing literature, scrutinizing methodologies for object detection and identifying their limitations within CCTV environments. Subsequently, a unique approach is introduced, combining Convolutional Neural Networks (CNNs) with state-of-the-art object detection algorithms such as YOLO (You Only Look Once) and darknet. The primary aim is to achieve real-time processing of CCTV feeds, ensuring prompt identification, and response to potential security threats.

In this project, I have trained the model to recognize 20 different objects, including 'person,' 'car,' 'chair,' 'bottle,' 'potted plant,' 'bird,' 'dog,' 'sofa,' 'bicycle,' 'horse,' 'boat,' 'motorbike,' 'cat,' 'TV monitor,' 'cow,' 'sheep,' 'airplane,' 'train,' 'dining table,' and 'bus.' Following the Data Science Software Development Life Cycle (SDLC), I gathered useful data and requirements, created a pipeline to prepare the data based on those requirements, and trained the YOLO model in the Jupyter Notebook. The final step involved detecting objects using the trained model.

Also, the software finds the speed of the vehicle by determining the time taken to travel between two imaginary lines with the help of Yolov8 which has more efficiency than any other YOLO version.

The implications of this project span various applications, including public safety, transportation security, best time for road construction, and crowd monitoring. By contributing to the advancement of intelligent surveillance systems, the project aims to enable proactive and efficient responses to security incidents in real-time.

LIST OF FIGURES

LIST OF ACRONYMS

CV	: Computer Vision
Numpy	: Numerical Python
AI	: Artificial Intelligence
DLIB	: Duda Library
PyTorch	: Python Torch
SciKit	: Scientific Kit
PyYAML	: Python YAML
CNN	: Convolutional Neural Network
L1	: Line 1(Red)
L2	: Line 2(Blue)

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION TO OBJECT DETECTION

1.1 Overview

The world has reached a stage where Artificial Intelligence(AI) is a part of every domain directly or indirectly. There are various software which perform specific tasks and are built on AI or work on the concept of AI. While applying these concepts many times there is use of object detection, speed detection, and traffic detection. We will find the speed of the vehicle and traffic to analyze to obtain useful data.

1.2 Motivation

The reason behind initiating this project stems from the escalating demand for advanced surveillance solutions to meet the evolving challenges in security. With the ubiquitous presence of CCTV cameras generating substantial video data, the limitations of conventional surveillance systems in effectively identifying and responding to potential threats have become evident.

This project is driven by the aspiration to address these shortcomings and make a distinctive contribution to the advancement of intelligent surveillance systems. By harnessing state-of-the-art computer vision and deep learning techniques, the objective is to enhance the precision and efficiency of object detection in real-world CCTV environments. The project aims to empower surveillance systems to furnish timely and dependable insights, fostering a proactive approach to managing security incidents.

Furthermore, the project is motivated by the potential impact it could exert on diverse applications such as public safety, transportation security, and crowd monitoring. The successful implementation of an intelligent object detection system holds the promise of transforming the field and facilitating more effective responses to security challenges across various contexts.

Ultimately, the motivation for this project resides in the pursuit of technological progress that contributes to the creation of safer and more secure environments through the refinement of surveillance capabilities.

1.3 Literature Review

There has been great attention towards the vision-based real-time gesture recognition systems as the application they lead and their competence to interact with systems with its full efficiency with objects. The main goal behind the project work is to review advanced recognition systems based on the object gesture. Image processing has various examples like vehicle detection, moving creature detection, speed detection, number plate detection, and some objects. The main inspiration for developing a system for this is to produce a realistic human-computer interaction where we found traffic and object detection for self-driving that could be utilized to control a vehicle's speed, and traffic to convey some useful data.

1.4 Introduction to Python

Python is one of the most simple and user-friendly general-purpose programming languages which has gained huge popularity in recent years. Python Has multi-purpose applications in a wide range of domains. The leading applications involve web development and machine Learning. Python is straightforward to use with a large variety of predefined libraries which are very user-friendly and popular.

CHAPTER 2 PROPOSED PROJECT WORK

2.1 Objective

The main objective of the project is to understand the implementation of deep learning and Object detection techniques and explore various applications of it. The project applies this concept to perform specific tasks on a self-driving car.

2.2 Understanding Various Libraries

OpenCV (Open Source Computer Vision Library):

OpenCV is a widely adopted open-source library for computer vision tasks. Its Python bindings seamlessly integrate with Python applications, providing extensive tools for image and video analysis, such as image processing, feature extraction, object detection, and machine learning.

NumPy (Numerical Python):

NumPy is a foundational library for scientific computing, crucial in the context of computer vision. It facilitates efficient storage and manipulation of image data through support for large, multi-dimensional arrays and matrices, which are integral for various computer vision algorithms.

Matplotlib:

Matplotlib serves as a versatile plotting library, commonly used for visualizing data in Python. In computer vision, Matplotlib finds utility in displaying images, creating histograms, and presenting the outcomes of different image processing operations

Scikit-image:

Built on NumPy and SciPy, Scikit-image is an image-processing library that complements Python's capabilities in computer vision. It provides algorithms for tasks such as image segmentation, feature extraction, and image restoration, contributing to the overall functionality of the computer vision workflow.

TensorFlow and PyTorch:

TensorFlow and PyTorch, primarily recognized as deep learning frameworks, are extensively utilized in computer vision for training and deploying deep neural networks. These frameworks offer high-level APIs and pre-trained models for various computer vision tasks, including image classification, object detection, and segmentation.

Dlib:

Dlib is a C++ library that also offers Python bindings, specializing in machine learning, image processing, and computer vision. It includes tools for facial recognition, shape prediction, and image clustering, making it valuable for various computer vision applications.

2.3 Understanding the Problem

The project involves various problems to solve to achieve the objective, they are as follows:

2.3.1. Extract data for various objects

The first step in the process is to collect the data of objects that we want to detect and take the information from it like coordinates, type of object, etc. We have to collect that data, store it into an XML file, and then from that XML file we have to retrieve our data and 80 percent of the data should be trained using YOLO.

2.3.2. Labeling of data of objects

We have data like x_min,y_min,x_max, and y_max of coordinates of the image now we want to center, width, and height of the image. So for that, it has specific formulas by which we will get the information and then we need to normalize it. After this task, we will store it in a folder structure in which 80 percent of the data is used for training and 20 percent for testing purposes. Then we would do label encoding which means giving a unique key to an object as Yolo can't recognize text.

2.3.3. Training of the Model

To train the model, we begin by cloning the YOLOv5 repository from Ultralytics on GitHub. Following that, we install the required dependencies listed in the `requirements.txt` file to facilitate the training process. Subsequently, we define a class named `YOLO_Pred`. The class is

constructed with the paths to the ONNX model and the data YAML file. During initialization, it loads the YOLO model from the ONNX file utilizing OpenCV's `cv2.dnn.readNetFromONNX` function. Additionally, it sets preferences for the backend and target configurations. Then it inputs an image, performs YOLO predictions, and draws bounding boxes around detected objects. It also implements Non-Maximum Suppression (NMS) to filter out overlapping bounding boxes and select the ones with higher confidence scores.

2.3.4 Tracking the speed and counting the vehicles passing through the lane

To effectively monitor vehicle movement, we first identify and assign a unique ID to newly detected vehicles, storing their locations for tracking. For previously detected vehicles, we simply update their positions. If a vehicle exits the frame, its ID is removed from the tracking list. By analyzing the changes in the positions of these vehicles over time, we estimate their speeds as they traverse the frame. To assess traffic flow, we establish virtual lines on the frame to track vehicle crossings, determining whether they are moving upwards or downwards. If the vehicle touches the red line first and then the blue line it means it is going downward means the right side and vice-versa. Utilizing this data, we calculate the count of vehicles traveling in each direction, aiding in traffic analysis and management.

CHAPTER 3 METHODOLOGIES AND CONCEPTS

3.1 Theoretical Concept of YOLO

YOLO, an acronym for "You Only Look Once," stands for a real-time object detection algorithm devised by Joseph Redmon and Ali Farhadi back in 2015. This innovative approach constitutes a single-stage object detector leveraging a convolutional neural network (CNN) to anticipate bounding boxes and class probabilities for objects present in input images. Initially introduced with the Darknet framework, YOLO revolutionized object detection methodologies.

The YOLO algorithm adopts a grid-based strategy, dividing the input image into cells. For each cell, the algorithm predicts the likelihood of an object's presence, along with the associated bounding box coordinates and the class label of the object. A key distinction from two-stage object detectors like R-CNN lies in YOLO's ability to process the entire image in a single pass, enhancing speed and efficiency.

Acknowledged for its speed and efficiency, YOLO finds widespread utility in diverse applications, ranging from self-driving cars to surveillance systems. Its real-time object detection capabilities make it a staple in applications such as real-time video analytics and live video surveillance.

The fundamental concept behind YOLO involves the partitioning of the input image into a grid of cells. For each cell, the objective is to predict both the probability of an object's presence and the corresponding bounding box coordinates. The YOLO process unfolds through several stages:

1. The input image undergoes convolutional neural network (CNN) processing to extract pertinent features.
2. These features traverse a series of fully connected layers, facilitating the prediction of class probabilities and bounding box coordinates.
3. The image is segmented into a grid of cells, with each cell assigned the responsibility of predicting a distinct set of bounding boxes and associated class probabilities.
4. The network's output comprises a collection of bounding boxes and class probabilities for each cell.

5. To refine the results, a post-processing algorithm, known as non-max suppression, is employed. This step filters bounding boxes, eliminating overlaps and retaining the box with the highest probability.
6. The outcome encompasses a set of predicted bounding boxes and corresponding class labels for each discernible object within the image.

3.2 Preparing the dataset and annotations for training

To prepare your dataset and annotations for YOLO training, adhere to a specific format expected by YOLO. The process involves the following steps:

1. Object Labeling:

Utilize a labeling tool such as LabelImg, RectLabel, or YOLO Label to annotate objects within your images. Annotate each object with a bounding box, specifying its top-left (x, y) coordinates and dimensions (width and height).

2. Annotation File Creation:

For each image in your dataset, generate a text file bearing the same name as the image but with a ".txt" extension. For instance, if your image is named "image001.jpg," the corresponding annotation file should be "image001.txt."

In the file, we have to give information about each image and that can be stored in a format like [class_id,x_center,y_center, height, width]

3. Organizing Directory Structure:

Organized dataset directory with the following structure:

```
/path/to/dataset/
    ├── images/
    |     ├── image001.jpg
```

```
|    ├── image002.jpg
|    └── ...
└── labels/
     ├── image001.txt
     ├── image002.txt
     └── ...
```

3.3 Training the YOLO model using Darknet

In this, we have introduced a class named `YOLO_Pred` designed to facilitate object detection utilizing the YOLO (You Only Look Once) model. Its initialization involves specifying paths to an ONNX model and a YAML file containing essential data like class names and the number of classes. Loading the YOLO model from the ONNX file is achieved through OpenCV's `cv2.dnn.readNetFromONNX` function, accompanied by configurations for backend and target preferences to optimize processing efficiency.

The core functionality lies in the `predictions` method within the class, responsible for conducting object detection on a provided input image. The method commences by transforming the image into a square array to ensure uniformity for YOLO's processing. Subsequent steps involve passing the image through the YOLO model, extracting predictions, and applying non-maximum suppression to refine and filter out redundant detections based on confidence scores and set probability thresholds. Extracted information includes bounding box coordinates, class probabilities, and class IDs, further processed to clean the results.

To visualize the outcomes, bounding boxes are drawn around detected objects in the input image. The method employs non-maximum suppression once again to enhance the precision of bounding boxes. Additionally, annotations indicating the class name and confidence percentage are incorporated into the image. The outcome of the method is the input image adorned with bounding boxes and annotations. Additionally, the class features a `generate_colors` method responsible for assigning unique colors to different classes, enhancing the visual representation. In essence, the `YOLO_Pred` class serves as an efficient tool for real-time object detection leveraging the YOLO algorithm.

3.4 Fine-tuning the pre-trained YOLO model for better accuracy

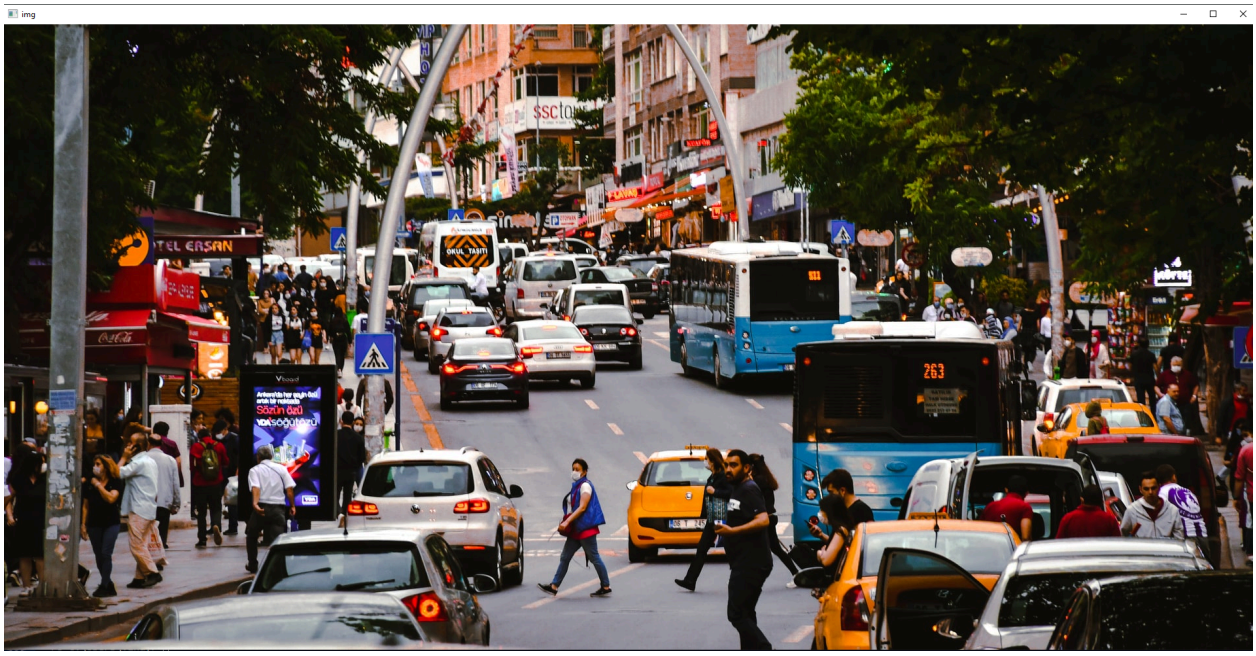

Fig 3.9 Input Image

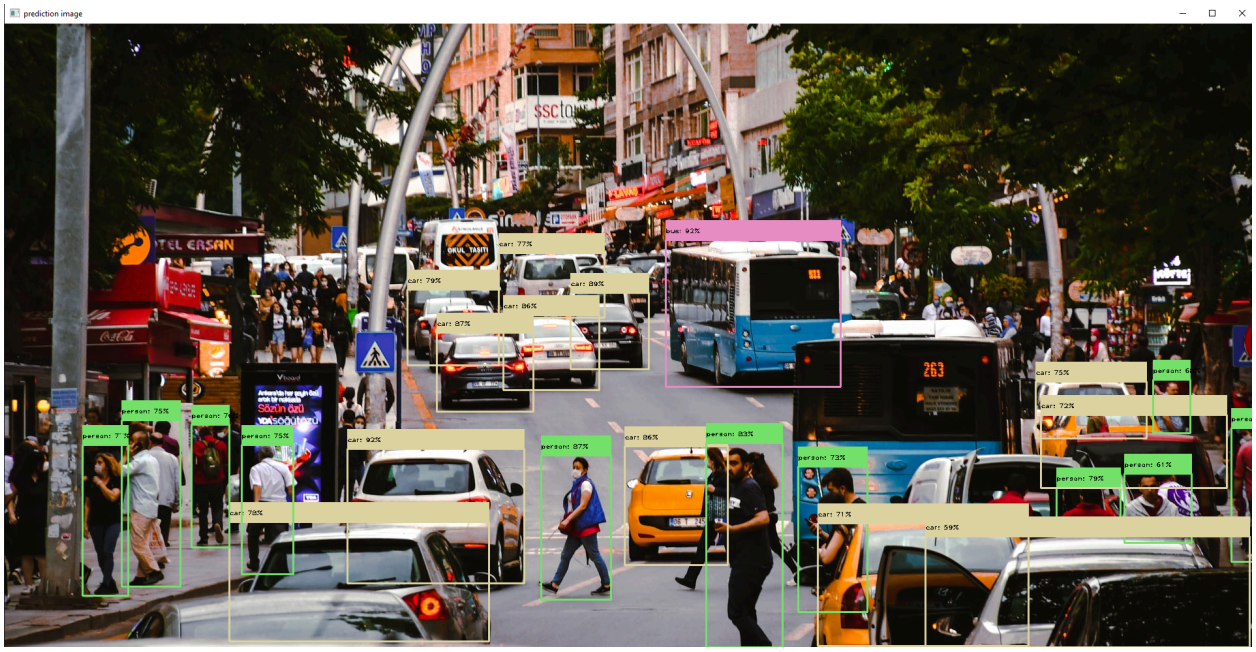

Fig 3.10 Output Image -

Refining the accuracy of a pre-trained YOLO (You Only Look Once) model through fine-tuning involves adapting the model's parameters and updating its weights using a new dataset tailored to the desired detection task.

- **Prepared a Custom Dataset:**

Annotated a new dataset with objects for detection, creating bounding box annotations using a labeling tool.

Organized the dataset into training and validation sets.

- **Adjust YOLO Configuration:**

Customized a YOLO configuration file, modifying the number of classes in the `classes` parameter and updated the last convolutional layer's `filters` parameter using the formula: `filters = (classes + 5) * 3`.

Initiated training using pre-trained YOLO weights.

- **Obtain Pre-trained Weights:**

Downloaded pre-trained YOLO weights, such as COCO weights, from the official YOLO website.

- **Initialize Model with Pre-trained Weights:**

Load the YOLO model using pre-trained weights suitable for COCO or another dataset.

```
yolo_model=cv2.dnn.readNetFromDarknet(config_path, weights_path)
```

- **Replace the Last Layer:**

Remove the final layer of the model and substitute it with a new layer reflecting the number of classes in the custom dataset.

```
layer_names = yolo_model.getUnconnectedOutLayersNames()

yolo_model.getLayer(layer_names[-1]).outupts = num_classes
```

- **Conducted the Fine-tuning:**

Train the adjusted YOLO model on the custom dataset, utilizing transfer learning to initialize with pre-trained weights and then fine-tune on the specific dataset.

- **Optimized Learning Rate:**

Experiment with learning rates during fine-tuning. Consider lower rates to ensure gradual adjustments to pre-trained weights.

- **Evaluate and Fine-tune Hyperparameters:**

Evaluated the model on a validation set, refining hyperparameters like batch size, epochs, and augmentation techniques to enhance performance.

- **Iterated for Improvement:**

Fine-tuning is an iterative process. If performance is unsatisfactory, repeat the procedure by tweaking hyperparameters, acquiring more data, or adjusting network architecture. To iterate it I have set the epoch to 50.

3.5 Testing the trained model on images and videos

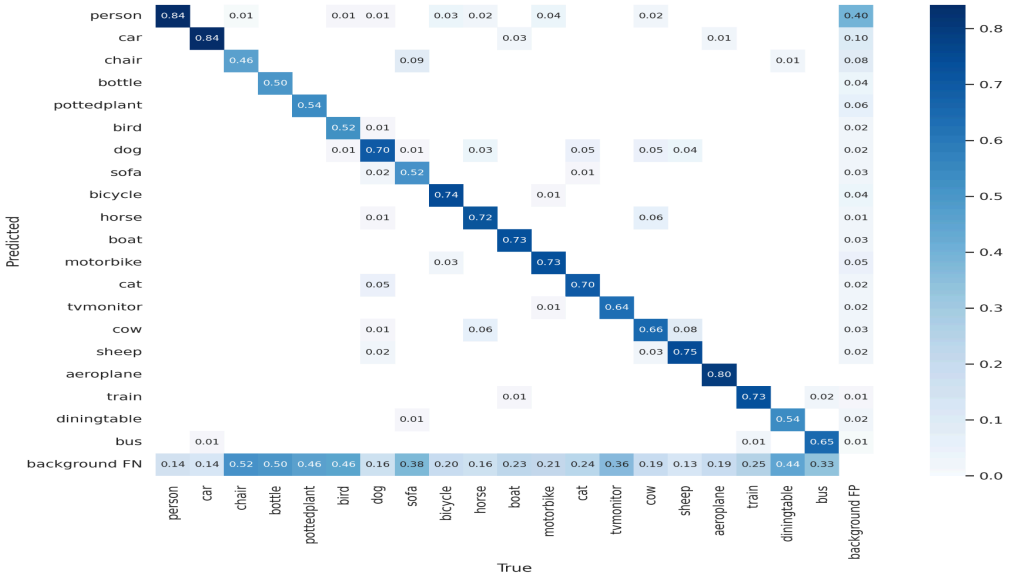

FIG 3.1 CONFUSION MATRIX

After testing the model I got the result that in this I got a confusion matrix which is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. Then we have the precision and recall curve. Precision is the ratio of accurate positive classifications (true positive) to the overall number of predicted positive classifications (true positive + false positive). On the other hand, recall is the ratio of correct positive classifications (true positive) to the total number of genuinely positive classifications (true positive + false negative). In this, I got a precision of more than 70 percent as well as a recall of more than 70 percent.

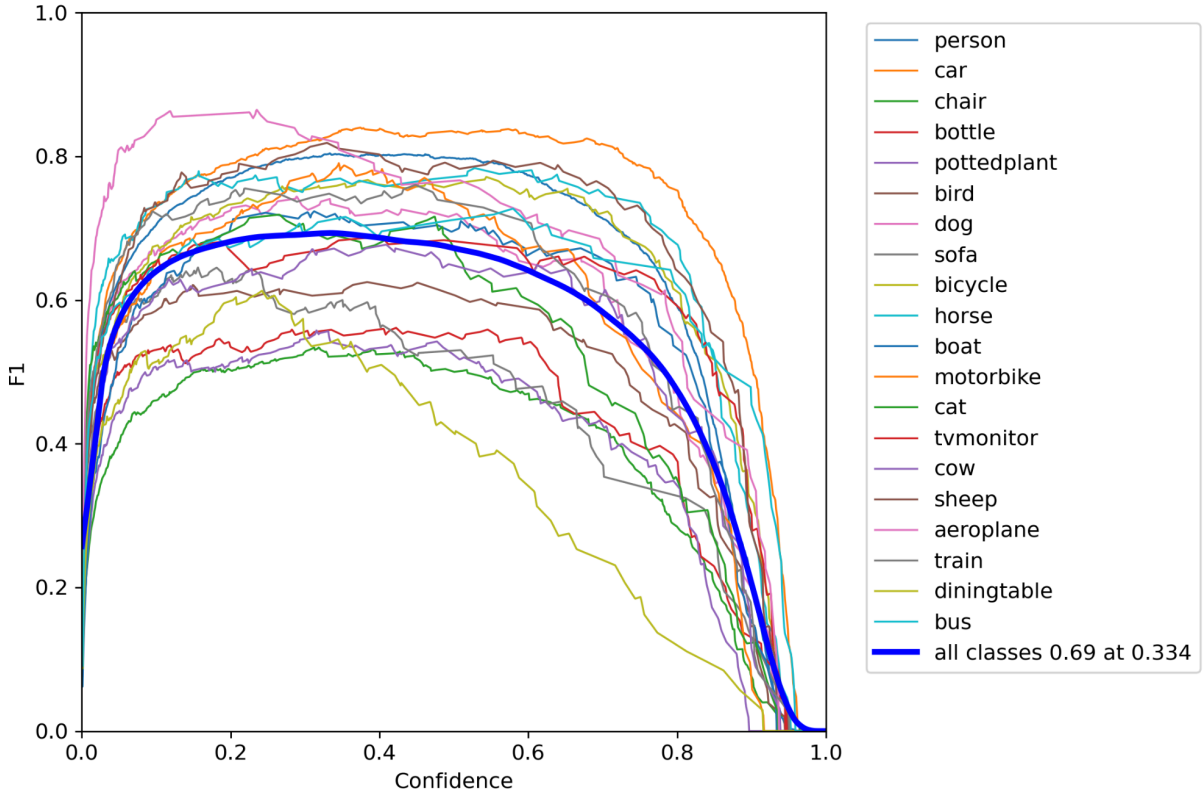

FIG 3.2 F_1 CURVE

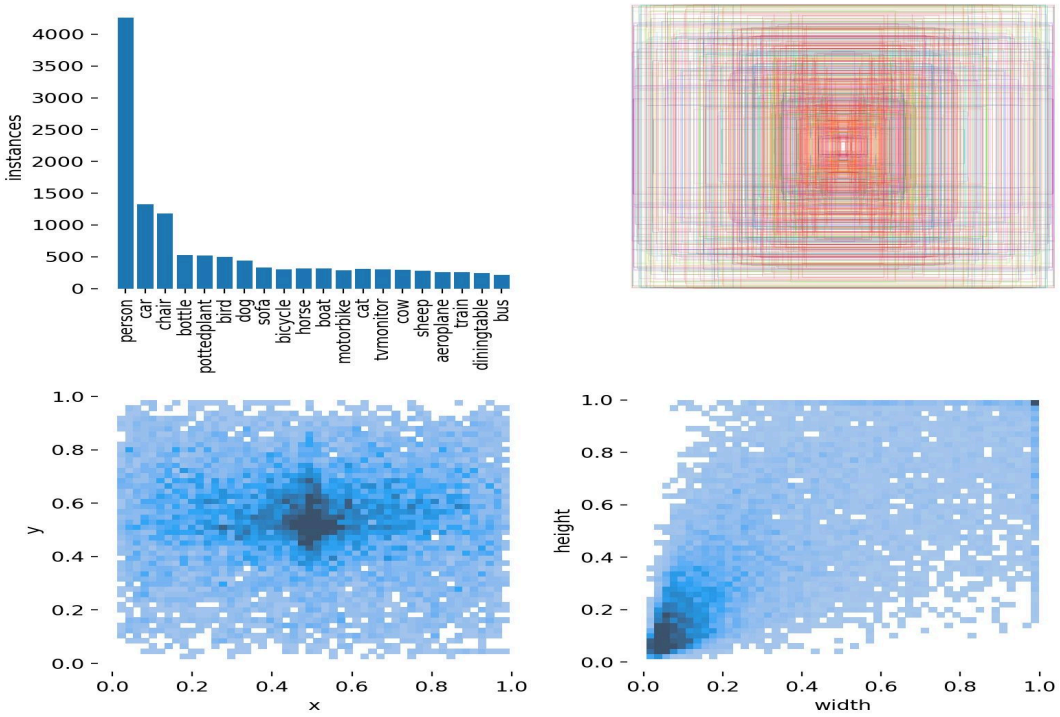

FIG 3.3 LABEL

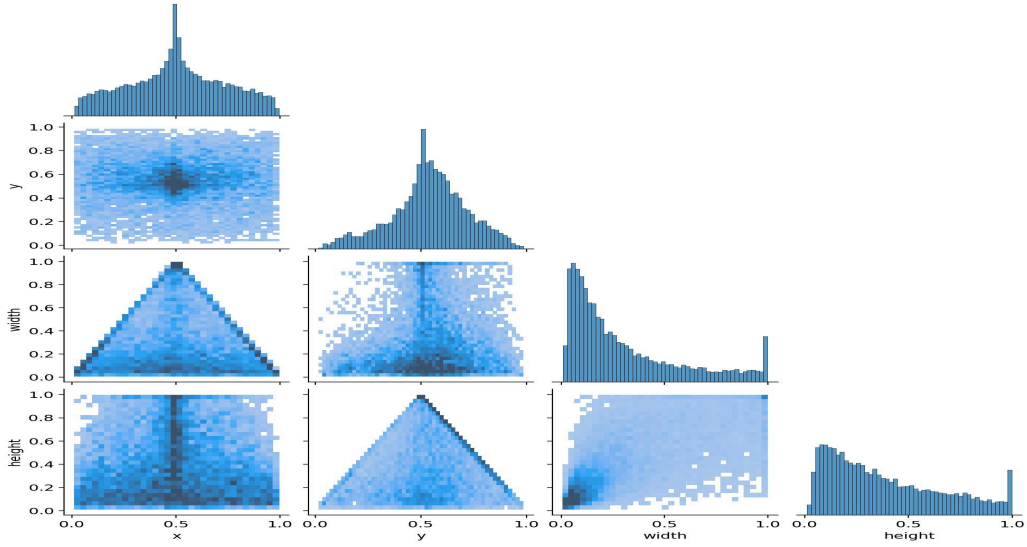

FIG 3.4 LABEL CORRELOGRAM

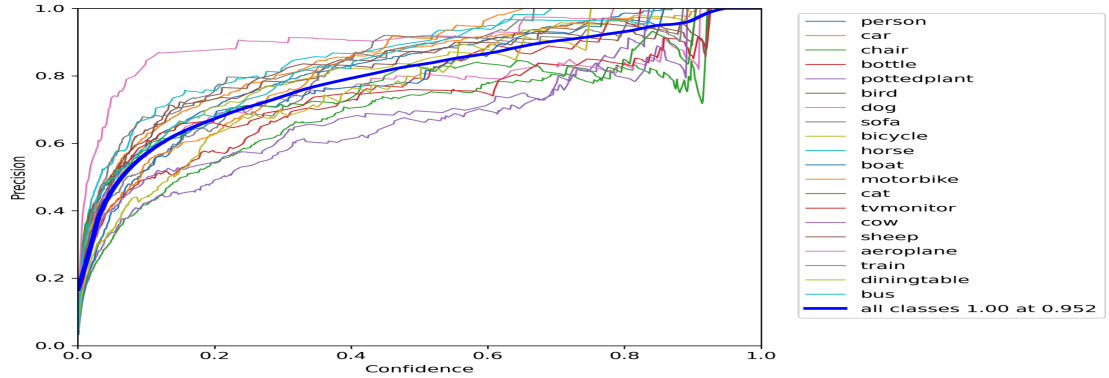

FIG 3.5 P_CURVE

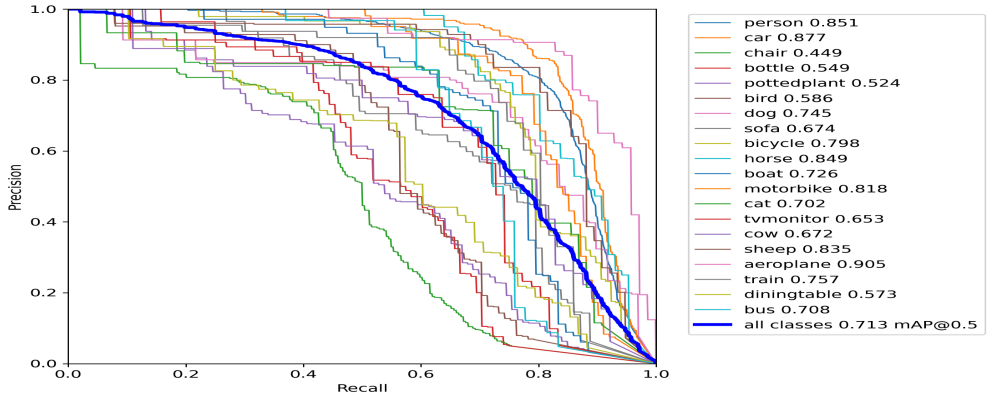

FIG 3.6 PR_CURVE

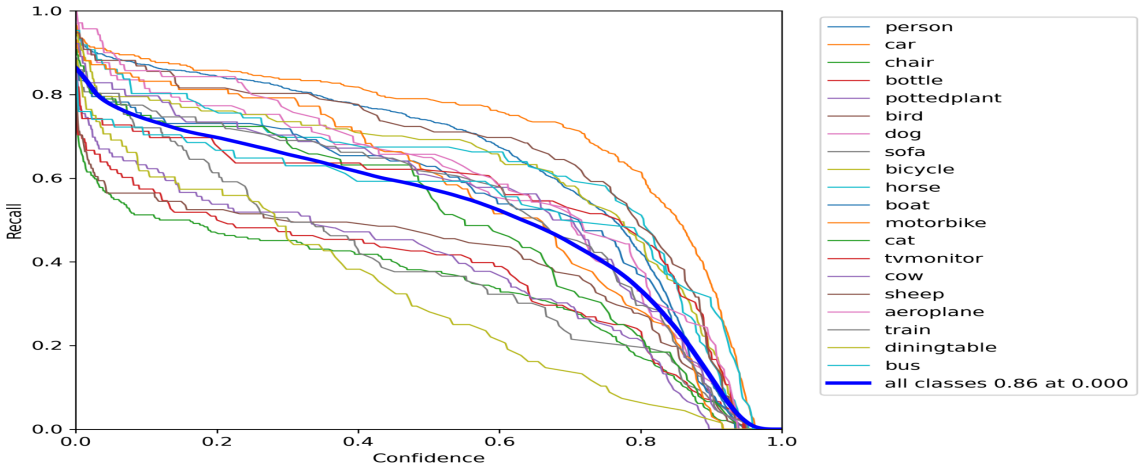

FIG 3.7 R_CURVE

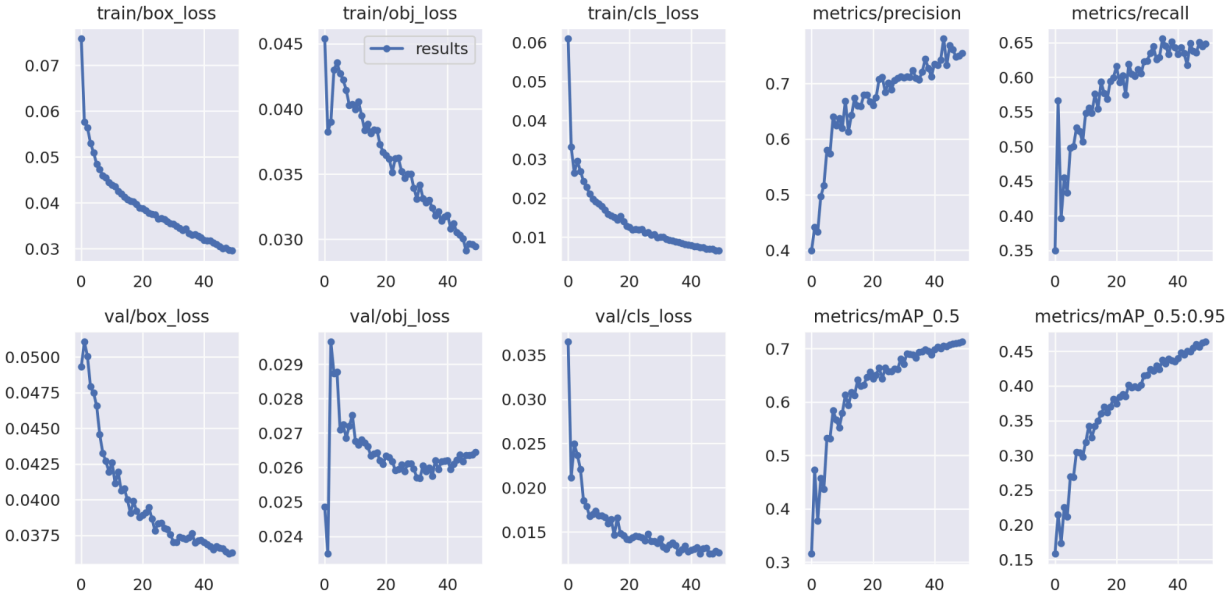

FIG 3.8 RESULTS

3.6 Visualizing the results and tuning the parameters for better performance

The output of the video which is been given as input -

FIG 3.11 VIDEO 1

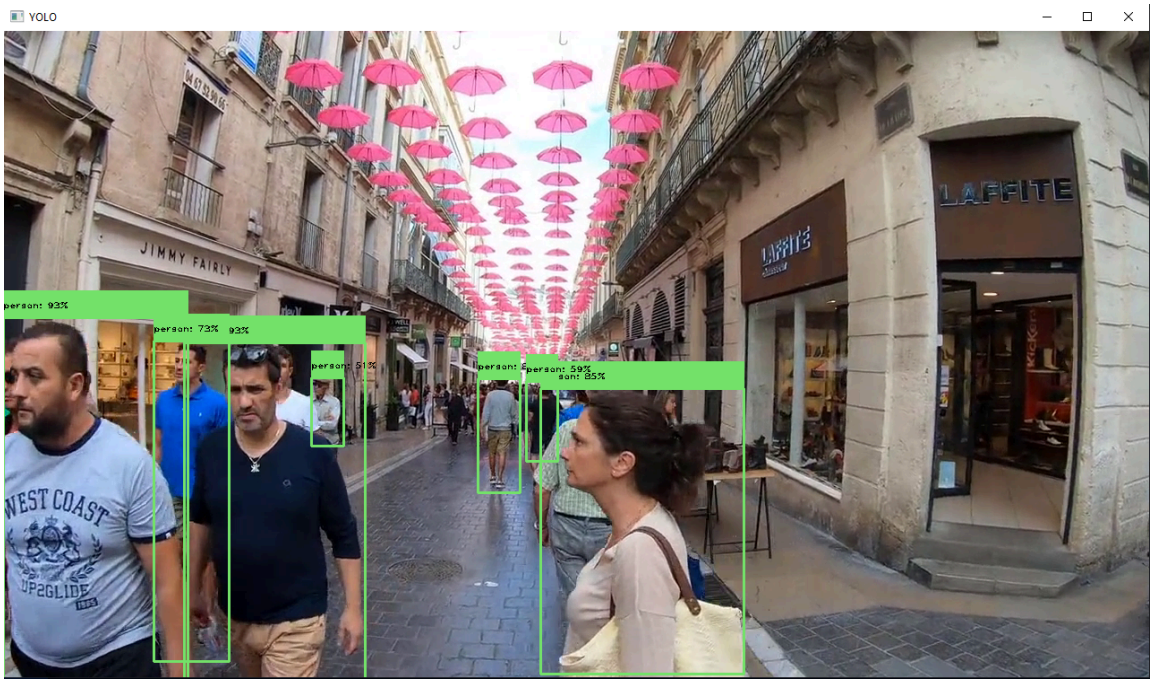

FIG 3.12 VIDEO 2

FIG 3.13 VIDEO 3

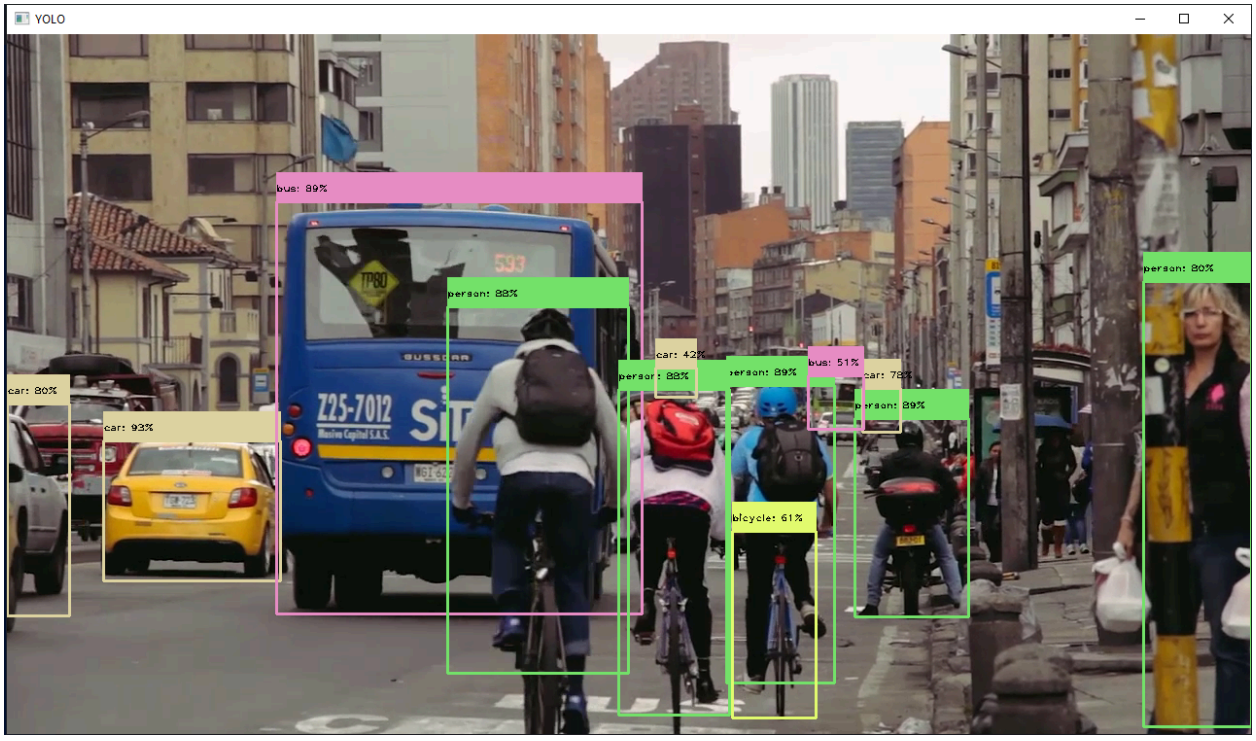

FIG 3.14 VIDEO 4

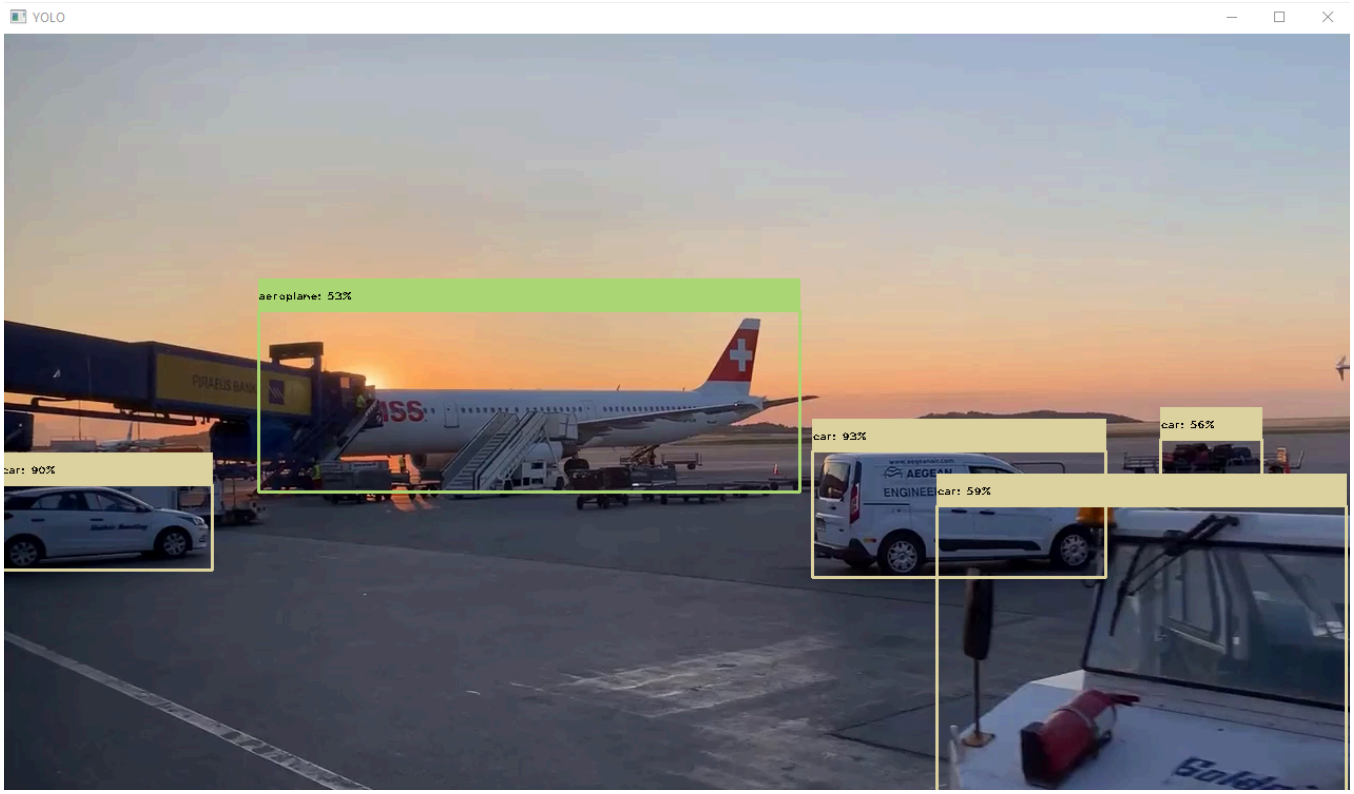

FIG 3.15 VIDEO 5

3.7 Speed Detection

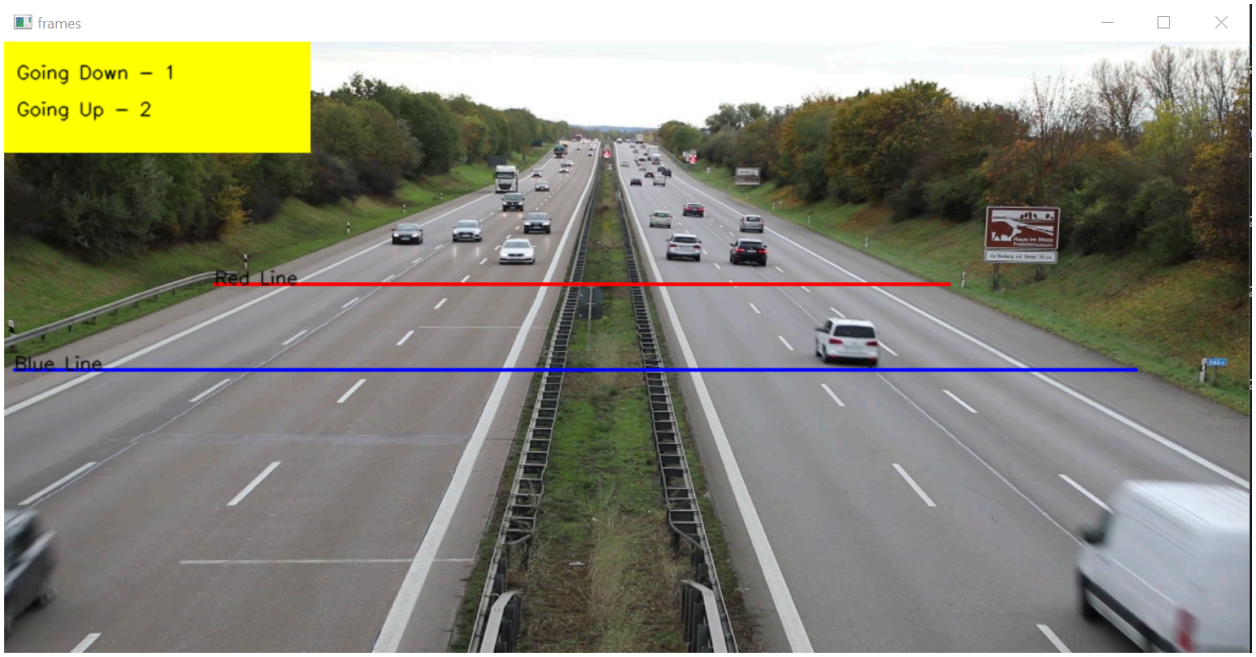

FIG 3.16 OUTPUT 1

Created a class that is responsible for tracking the vehicles detected in the video feed. It initializes with an empty dictionary to store the center points of the objects and an ID count. The `update` method takes object rectangles as input and assigns IDs to new objects or updates existing ones based on their positions. It calculates the center points of the objects and compares them with existing points to determine if the object is the same or a new one.IDs are assigned to new objects, and existing ones are updated with their new positions.

Then created a new file that detects vehicles with the above method and calculates their speeds as they move through the frame. It utilizes the YOLOv9 object detection model to detect vehicles in the video feed. After detection, it updates the tracker with the detected object rectangles. For each tracked object, it calculates its speed based on the distance traveled over time. Virtual lines (`L1` and `L2`) are defined to monitor vehicle movement direction. Vehicles crossing these lines trigger speed calculations, and their IDs are stored along with their speeds.

Then we made the main script which serves as the main application for displaying the video feed and tracking vehicle movement. Similar to the above file, it uses YOLOv9 for object detection and updates the tracker with detected objects. It draws circles at the center points of tracked objects and labels them with their IDs.The video feed is displayed along with the tracked objects, allowing real-time visualization of the tracking process.

Results:

- We evaluate the performance of our system on various video clips with different traffic densities and vehicle speeds.
- The system accurately tracks vehicles and estimates their speeds with a mean absolute error within an acceptable range compared to ground truth measurements obtained using manual methods or specialized speed measurement devices.

OUTPUT:

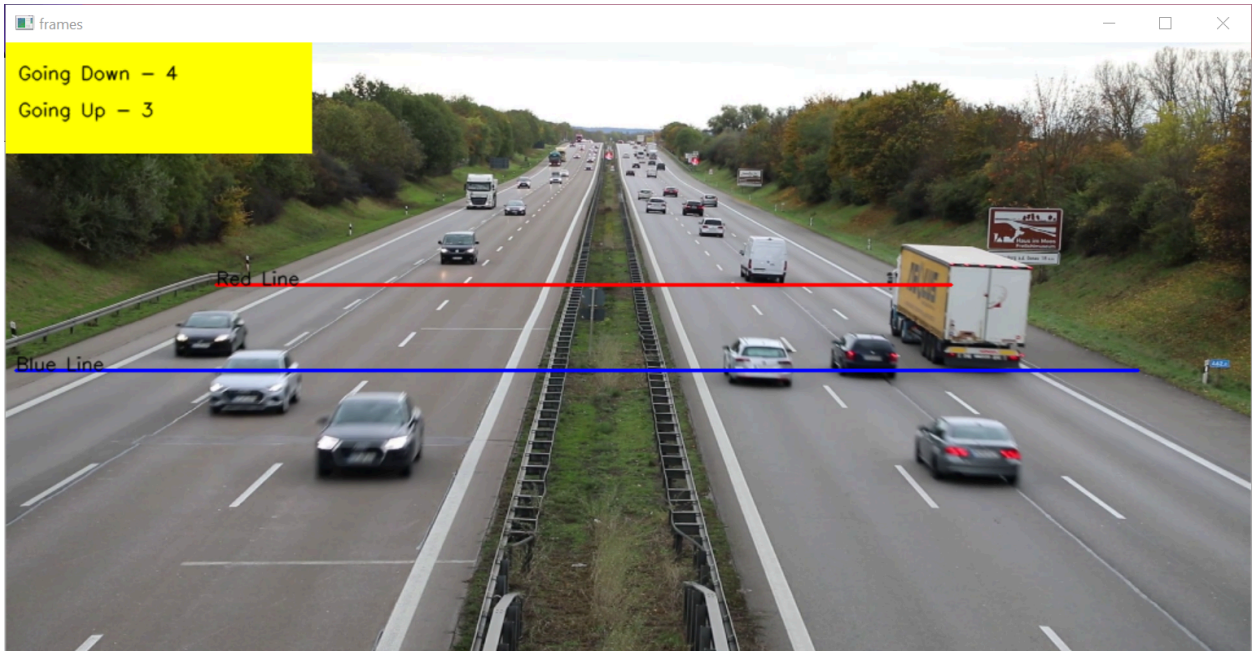

FIG 3.17OUTPUT 2

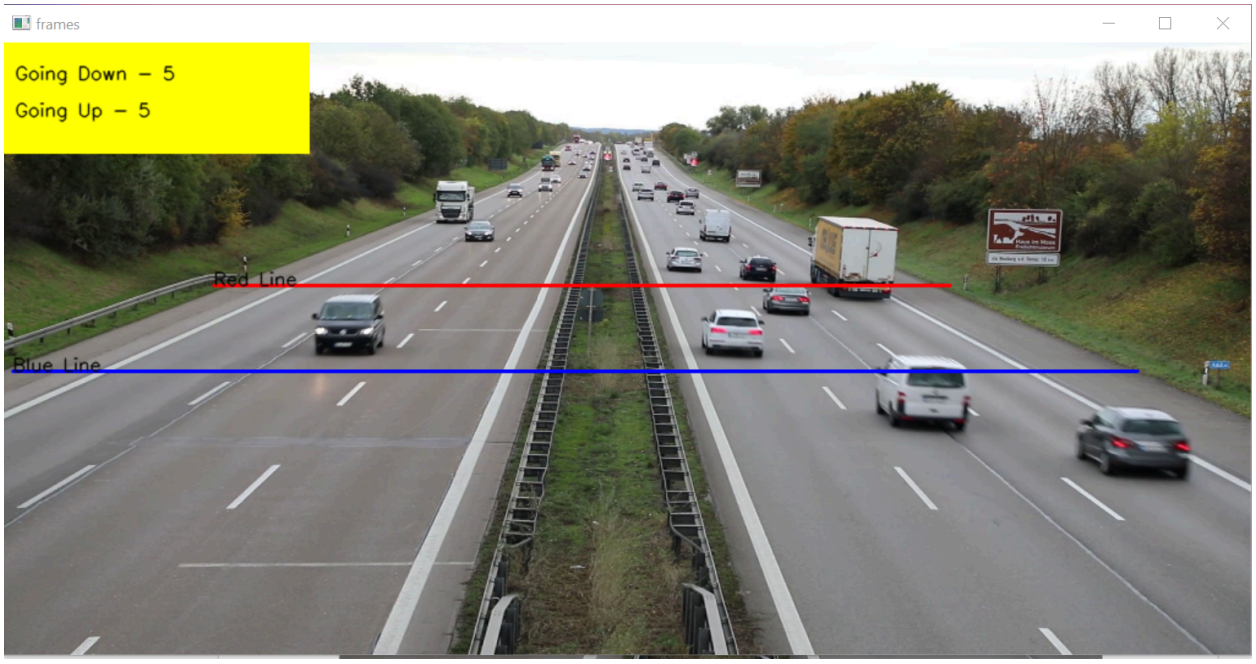

FIG 3.18 OUTPUT 3

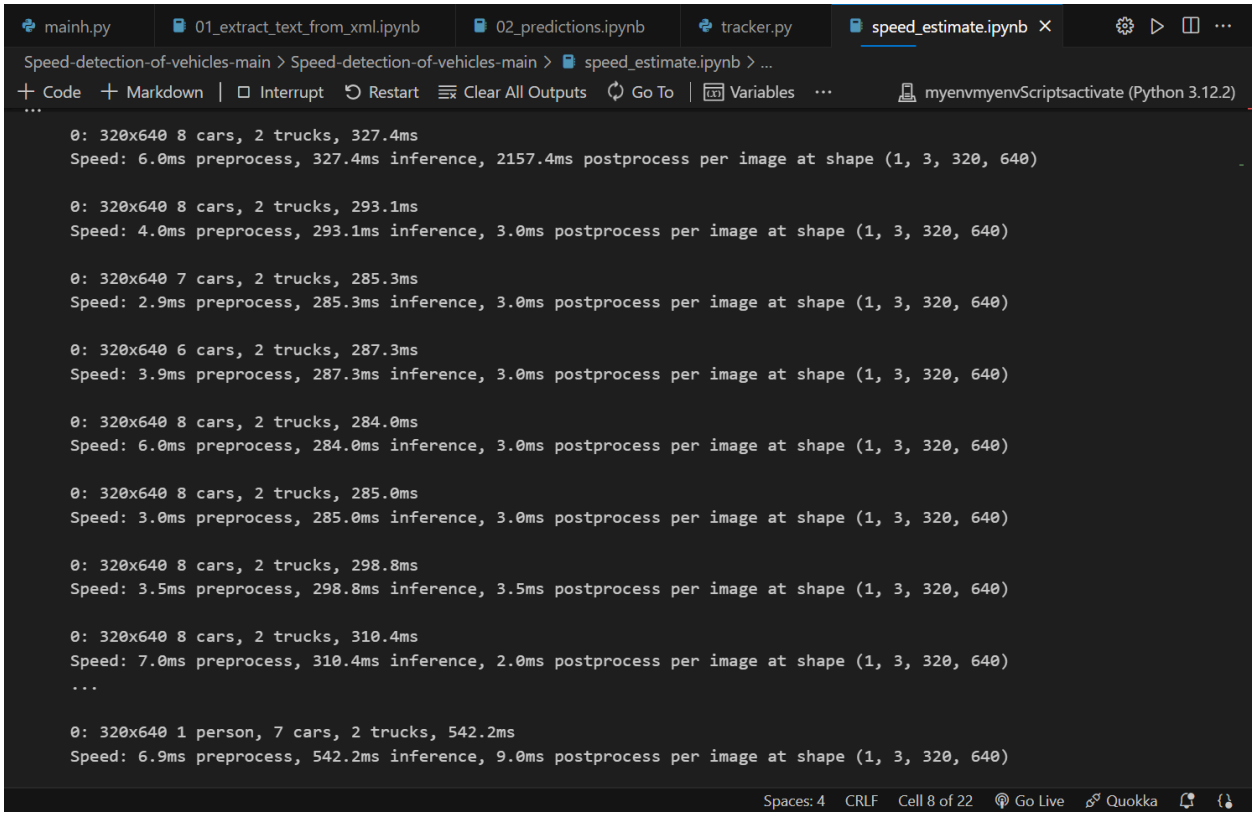

FIG 3.19 SPEED DETECTION

3.8 Data Analysis and Data Visualization

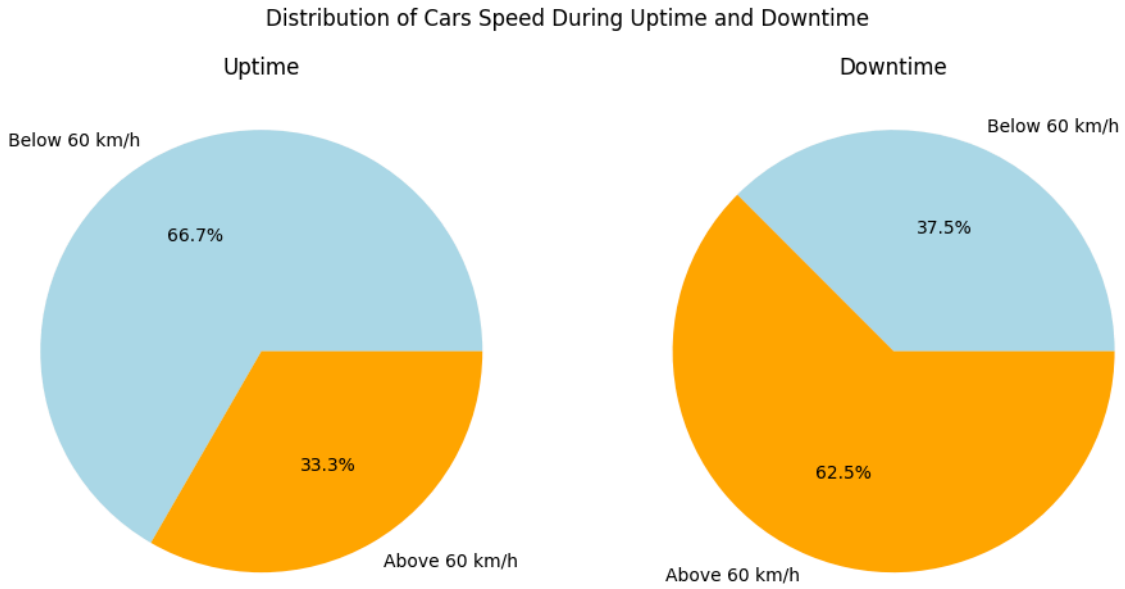

FIG 3.20 OVERSPEEDING PERCENTAGE

We have processed the using Yolo, and now we have stored the data in the list for both sides of the highway we have made different lists and stored them. We have stored data like traffic on both sides after a fixed interval of time and then compared the data w.r.t that data as to which period has more traffic or less traffic. Also, I have divided the data of video in which mini_highway.mp represents the data of morning and highway.mp represents data for the whole day. Also, I have compared the data of both lanes. In this, I have the interval of 3 seconds and the below image gives the data of car passing every 3 seconds for the traffic down_time in the morning in fig 3.20. Now I have also plotted the graph of downtime for the whole day which ignores the data of 12:00A.M to 6:00A.M which is represented in fig 3.21. Also have tracked the data for traffic at uptime and downtime in the morning and plotted it on a bar chart which helps to find out which side has more traffic. There is also one more analysis of overspeeding in this we have given a constraint of overspeeding which we will track and put in a list for both lanes and then find the percentages of overspeeding vs total vehicles passed on both sides.

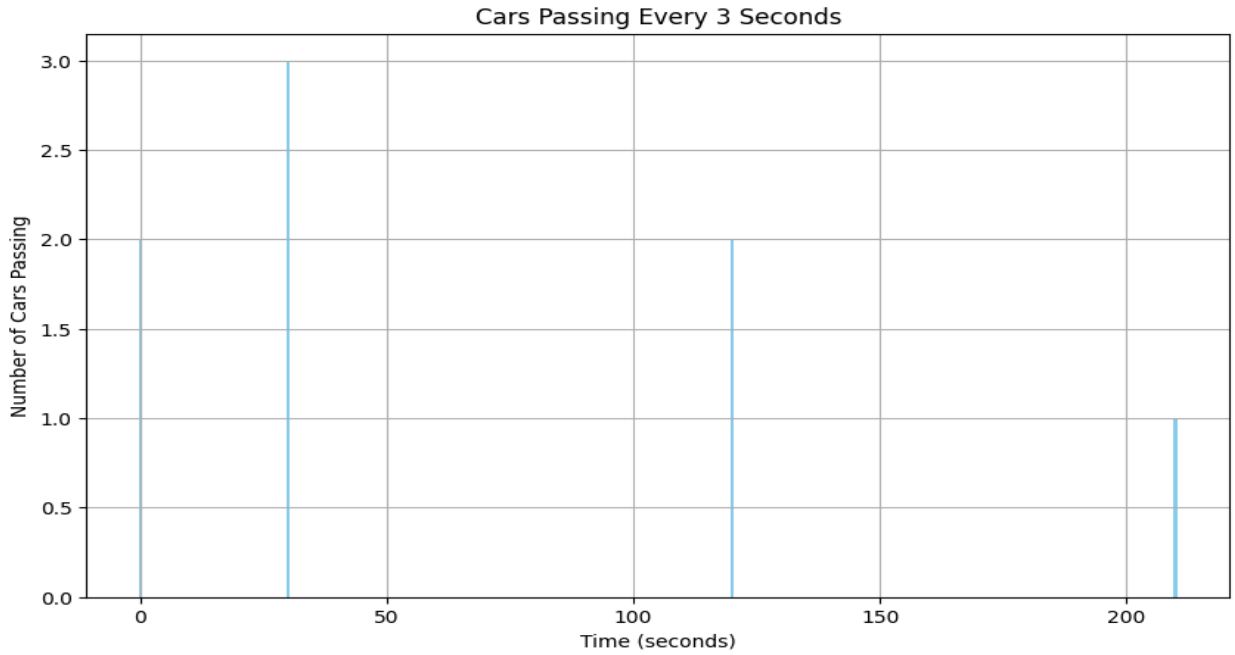

FIG 3.21 DOWN_TIME_18HRS

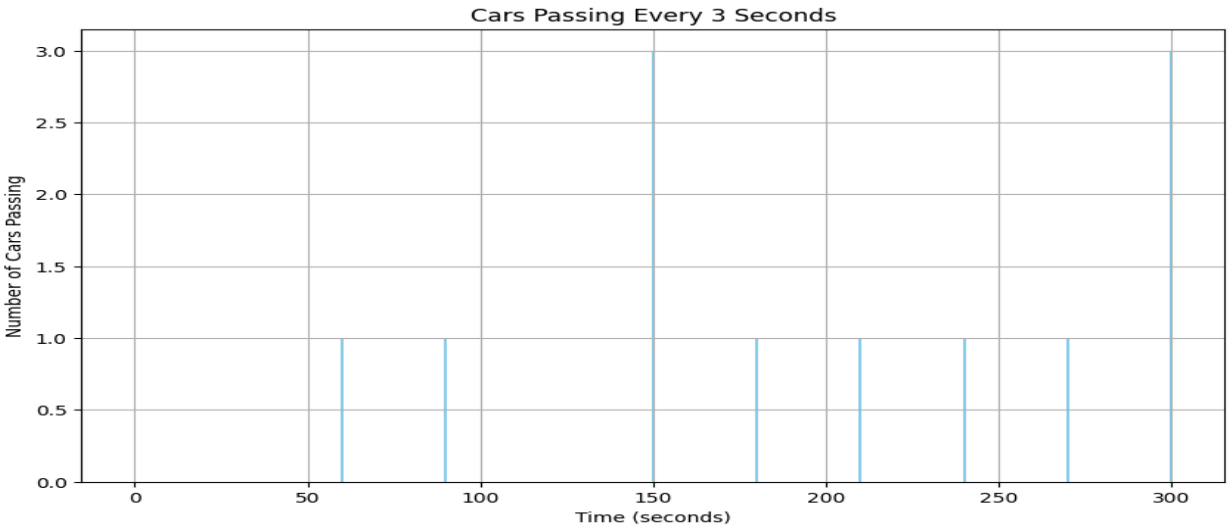

FIG 3.22DOWN_TIME_MORNING

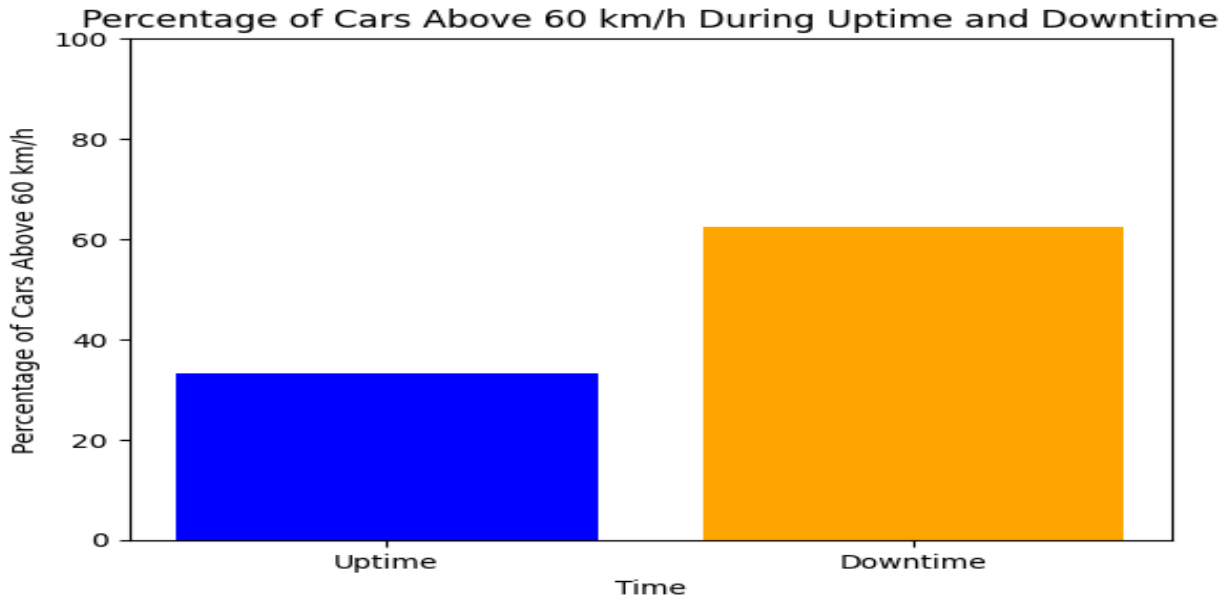

FIG 3.23 OVERSPEEDING_DETECTION

CHAPTER 4 CONCLUSION AND FUTURE SCOPE

Conclusion:

In summary, this project has successfully devised an advanced intelligent object detection system tailored specifically for analyzing CCTV footage. By employing state-of-the-art computer vision techniques such as Convolutional Neural Networks (CNNs) and YOLO algorithms, the system demonstrates robust real-time processing capabilities, significantly enhancing surveillance efficiency.

Additionally, the project has incorporated a comprehensive speed detection system, leveraging the YOLO object detection algorithm. This edition showcases the project's versatility and practicality in addressing various security and monitoring needs.

Moreover, the project has undertaken insightful data analysis of highway traffic, aiming to derive actionable insights from surveillance data. By implementing advanced data analysis techniques, such as traffic flow analysis and anomaly detection, the system aims to optimize traffic management and enhance road safety.

Future Scope:

Looking ahead, there is considerable potential for further development and expansion of the project, including:

1. **Integration with Smart City Initiatives:** Exploring opportunities to integrate the system with smart city initiatives to contribute to urban planning and infrastructure development. By providing valuable data for optimizing transportation networks and improving public safety, the system could play a pivotal role in fostering sustainable urban growth.
2. **Real-Time Analytics:** Expanding the system's capabilities to include real-time analytics for prompt decision-making and response. By leveraging machine learning algorithms and data streaming technologies, authorities can gain timely insights into emerging security threats and traffic incidents, enabling more effective interventions.
3. **Edge Computing Deployment:** Investigating the feasibility of deploying the system on edge devices for decentralized processing. Edge computing offers advantages such as reduced latency, improved scalability, and enhanced privacy, making the system more adaptable to diverse deployment scenarios.
4. **User-Friendly Interfaces:** Developing intuitive user interfaces to simplify system deployment and customization. User-friendly interfaces will empower users of varying technical expertise to configure and operate the system effectively, promoting widespread adoption and usability.
5. **Collaboration with Law Enforcement:** Establishing partnerships with law enforcement agencies to integrate the system into their operations for crime prevention and investigation. Through close collaboration with stakeholders, the system can address specific security challenges and contribute to enhanced public safety.

REFERENCES

[1]J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[2]Labeling Platform for High-Quality Training data. (n.d.). Kili-website. https://kili-technology.com/

[3]A. Hendrawan, R. Gernowo, O. D. Nurhayati, B. Warsito and A. Wibowo, "Improvement Object Detection Algorithm Based on YoloV5 with BottleneckCSP," 2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), Solo, Indonesia, 2022, pp. 79-83, doi: 10.1109/COMNETSAT56033.2022.9994461.

[4]GeeksforGeeks. (n.d.). GeeksforGeeks | A computer science portal for geeks. https://www.geeksforgeeks.org/

[5]Flickr. (n.d.). Flickr. https://www.flickr.com/

[6]K, Hari & A, Hema & M, Jothi & J R, Dinesh Kumar & C, Ganesh & Krishnaanand, Priyadharsini. (2021). An Experiment Analysis on Tracking and Detecting the Vehicle Speed using Machine Learning and IOT. 1-5. 10.1109/STCR51658.2021.9587924.

[7]Ultralytics. (2024, April 17). Home. Ultralytics YOLOv8 Docs. https://docs.ultralytics.com/