Automated traffic management system for emergency vehicles and two-wheeler rider helmet detection

Abstract

This abstract presents a comprehensive framework for real-time detection and recognition of Lightbars (LED)s, automated helmet usage, and license plate recognition using the You Only Look Once (YOLOV5) object detection algorithm. The proposed systemaims to enhance traffic management and safety by leveraging computer vision techniques to detect emergency vehicles, enforce traffic regulations, and identify potential traffic violations.

The first component of the system focuses on Lightbars (LED) detection. By employing the YOLO algorithm, the system can efficiently identify emergency vehicles equipped with Lightbars (LED)s in the model. This detection mechanism enables the timely recognition of emergency situations, facilitating rapid response times and minimizing traffic disruptions. Once a Lightbars (LED) is detected, the system initiates signal control to prioritize the passage of emergency vehicles.

Signal control is an essential feature of the proposed system. When a Lightbars (LED) is detected, the system automatically adjusts the traffic signals in the vicinity to ensure a green light for the approaching emergency vehicles. This enables efficient traffic flow andminimizes delays, significantly improving emergency response times. Once the emergencyvehicle has passed, the system reverts the signals to their regular operation. Furthermore,the system incorporates automated helmet detection for enhancing road safety. By employing YOLO, the system identifies motorcyclists and detects whether they are wearing helmets. This feature aids in enforcing helmet regulations, as it can identify non-compliant riders in real-time. Automated alerts can be sent to authorities, enabling themto take appropriate actions to ensure rider safety.

In conclusion, the proposed YOLO-based Lightbars (LED) detection, signal control, automated helmet detection, and license plate recognition system offers a comprehensive solution for enhancing traffic management and safety. By leveraging real- time computer vision capabilities, the system enables the timely detection of emergency vehicles, facilitates smooth traffic flow, enforces helmet regulations, and ensures compliance with traffic laws by identifying vehicles through license plate recognition. Theintegration of these functionalities into a single system provides a powerful tool for trafficauthorities, contributing to safer and more efficient roadways.

Contents

1. Introduction and Motivation

1.1 Introduction

Traffic congestion poses a significant challenge in urban areas worldwide, with detrimental effectson transportation efficiency and quality of life. In densely populated regions, such as India, the problem of traffic congestion is particularly pronounced, leading to increased travel times for commuters and compromised emergency response times. The consequences of delayed emergency services are grave, as every minute can make a difference in life-or-death situations.

The alarming statistics on road accidents and fatalities in India highlight the urgent need for effective traffic management solutions. According to reports, a substantial number of road accident victims lose their lives due to delayed ambulance services caused by traffic congestion. In addition, the lack of helmet usage among two-wheeler riders significantly contributes to road accident fatalities and injuries. These issues demand immediate attention and innovative approaches to ensure the safety and well-being of individuals on the road.

Figure 1.1.1- Lightbar (LED)

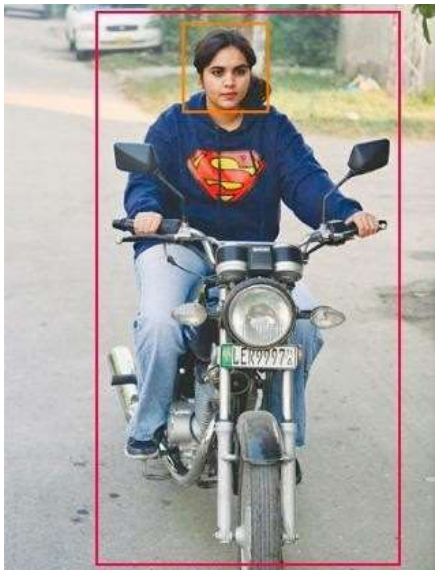


Figure 1.1.2 – Rider without helmet

In response to these pressing challenges, the project aims to design and implement a comprehensive traffic management system for emergency vehicles, coupled with helmet detection technology. By leveraging advancements in computer vision, machine learning, and intelligent algorithms, the system will enable detection of emergency vehicles and optimize

traffic signal control accordingly. This proactive approach will help emergency vehicles navigate through congested roads swiftly and safely, thereby reducing response times and potentially saving lives

The LED Light Bar Detection and Signal Control component addresses the issue of emergency vehicles facing delays and difficulties in reaching their destinations promptly. By leveraging computer vision algorithms, the system can accurately detect the presence of an ambulance or emergency vehicle by analyzing the unique light patterns emitted by their LED light bars. Upon detection, the system will initiate a signal control mechanism that activates a green light at nearby intersections, providing a clear and unobstructed path for the emergency vehicle. This real-time response will significantly improve the efficiency of emergency vehicle navigation and reduce response times, potentially saving lives in critical situations.

The Helmet Detection with License Plate Recording and Email Notification System component focuses on promoting safe riding practices for two-wheeler riders. Non-compliance with helmet regulations is a prevalent issue, leading to a higher risk of severe injuries and fatalities in accidents. Our system utilizes computer vision and machine learning algorithms to detect riders who are not wearing helmets in real-time.

By analyzing video streams from strategically placed cameras, the system can identify riders without helmets and capture their license plate information simultaneously. This data is then recorded and sent to a specific person or authority using an email API. This feature enables traffic authorities or concerned individuals to take appropriate actions, such as issuing fines or educating violators about the importance of helmet usage. By integrating license plate recording and email notifications, the system aims to improve accountability and ensure compliance with safety regulations.

In conclusion, our project aims to enhance road safety by implementing an intelligent system that addresses two critical aspects: LED light bar detection and signal control for ambulances, and helmet detection with license plate recording and email notifications for two-wheeler riders. By utilizing computer vision, machine learning, and communication technologies, we strive to improve emergency vehicle navigation, promote safe riding practices, and ultimately make our roads safer for everyone.

1.2 Literature Survey

Huansheng Song and Haoxiang Liang **[1]** devised a technique for a vision-based system that detects and counts vehicles. Initially, the highway road surface in the image undergoes extraction and segmentation into a distant area and a nearby area through a novel segmentation method, a crucial step for enhancing vehicle detection. Subsequently, both areas are input into the YOLOv3network to identify the vehicle type and its location.

K Agrawal et al. **[2]** introduced a vehicle detection and tracking system utilizing Image processingand Neural Network. The system recognizes vehicles by extracting features from different layers of CNN in the input image after a predefined stage. This approach boasts a shorter processing time compared to other methods. The framework takes traffic representations, encompassing various vehicle types and other road elements, as inputs. The resulting output highlights ambulances discerned from the traffic data.

Eshwar Prithvi and Pabitra Mohan **[3]** proposes a Vehicular Ad-Hoc Network (VANET) which usesvarious methods to detect emergency vehicles. It proposes using of RFID and OCR to clear the traffic.

J. Chiverton **[7]** he proposes a system which uses background subtraction to classify motorcycle riders and detects the helmet.

[14] Traffic Light Control in Non-Stationary Environments based on Multi agent Q-learning" by Abdoos M., Mozayani N and Bazzan A.L.C. explores a novel approach to traffic light control using multi-agent Q-learning. The authors model a large traffic network as a multi-agent system and use reinforcement learning techniques. They employ Q-learning, where the average queue length in approaching links is used to estimate states. The simulation results demonstrate that the proposed Q-learning outperformed the fixed time method under different traffic demands, providing a more adaptive and efficient method for traffic light control.

[17] "Automatic Helmet Detection in Real-Time and Surveillance Video" by Shubham Kumar. presents an automated system to identify motorcyclists without a helmet from traffic surveillance videos in real-time. The proposed model uses a single neural network per image, making it quicker than R-CNN and Fast R-CNN, which use multiple neural networks. The model was trained on the COCO dataset and aimed to detect and classify bike riders who are wearing a helmet as well as those who are not.

[26] Automatic Number Plate Recognition" discusses the development of an automatic number plate detection and recognition system for Indian vehicles. The proposed system first detects the vehicle and then captures the vehicle image. Vehicle number plate region is extracted using the

image segmentation and characters are recognized using optical character recognition technique.

[27] "Ambulance Detection System" discusses the development of a computer vision system to robustly detect ambulances observed from a static camera. Robust and reliable ambulance detection plays an important role for priority systems. These aim to find the shortest possible paths of ambulances till their destination by managing signaling networks with traffic lights.

[28] "Methods for extraction of features and discrimination of emergency sirens" by A. Otlora, D. Osorio, and N. Moreno presents the analysis, study, and tests, carried out to four different methods combined of extraction of characteristics and discrimination or identification of these, in order to determine an adequate and effective method of identification and discrimination of Emergency Sirens (Police, Ambulance, etc.) From the Cepstrum, MFCC's, FFT and Spectrogram as alternatives for extraction of characteristics of the signals to work, in addition Artificial Neural Networks (ANN) and Cross Correlation as discriminator and identifiers

"A Real-Time Safety Helmet Wearing Detection Approach Based on CSYOLOv3" by Haikuan Wang et al **[19]**. presents a novel safety helmet wearing detection model based on improved YOLOv3 (named CSYOLOv3) to enhance the capability of target detection on the construction site. The backbone network of darknet53 is improved by applying the cross stage partial network (CSPNet), which reduces the calculation cost and improves the training speed. The spatial pyramid pooling (SPP) structure is employed in the YOLOv3 model, and the multi-scale prediction network is improved by combining the top-down and bottom-up feature fusion strategies to realize the feature enhancement.

We use the above methods to detect the emergency vehicles in traffic and use a raspberry pi to trigger the traffic light to turn green until the vehicle is passed and the traffic light is reset to original timer and the methods discussed in "Automatic Helmet Detection in Real-Time and Surveillance Video", "A Real-Time Safety Helmet Wearing Detection Approach Based on CSYOLOv3", and "Deep Learning-Based Safety Helmet Detection in Engineering Management Based on Convolutional Neural Networks" can be utilized to implement the helmet detection feature for two-wheeler riders in our project. the principles outlined in "Automatic Number Plate Recognition" can guide the development of the license plate recording feature.

1.3 Motivation

The motivation behind this project lies in the profound impact it can have on improving traffic management, emergency response, and road safety. By developing an intelligent traffic management system specifically tailored for emergency vehicles, the project aims to address the critical issue of delayed ambulance services due to traffic congestion. Timely arrival of emergencymedical assistance can significantly increase the chances of survival and minimize the severity ofinjuries.

Furthermore, the project seeks to tackle the issue of non-compliance with helmet usage among two-wheeler riders. Wearing helmets is a proven safety measure that can prevent head injuries and save lives in the event of accidents. By integrating helmet detection technology into the traffic management system, the project aims to create awareness and encourage adherence to helmet-wearing regulations, leading to a reduction in the number of fatalities and severe injuries on the road.

Moreover, the project aligns with the broader goals of leveraging emerging technologies to address societal challenges. Computer vision, machine learning, and intelligent algorithms have witnessed significant advancements in recent years. By harnessing these technologies and applying them to the domain of traffic management, the project aims to demonstrate their practical application and potential to bring about positive change.

Ultimately, the motivation behind this project stems from a deep concern for public safety and a commitment to enhancing the efficiency and effectiveness of emergency services. By developing an innovative traffic management system that integrates emergency vehicle detection andhelmet detection, the project endeavors to create a safer road environment, improve emergency response times, and promote responsible road behavior. The potential benefits of this project are far-reaching, impacting not only individuals directly involved in accidents but also their families, communities, and society as a whole.

One of the key aspects of the project is the development of an intelligent traffic management system tailored specifically for emergency vehicles. Traffic congestion can significantly delay ambulance services, jeopardizing the lives of those in critical need of medical assistance. By leveraging real-time data, advanced algorithms, and predictive models, the system aims to optimize traffic flow, prioritize emergency vehicles, and provide them with the fastest and most efficient routes. This can help reduce response times, ultimately saving lives and minimizing the severity of injuries.

2.Background Theory

2.1 Introduction.

We use YOLO to detect helmets and emergency vehicles The detection of helmets on individuals using the YOLO (You Only Look Once) algorithm relies on object detection techniques. YOLO is a popular deep learning-based algorithm that achieves real-time object detection by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. The algorithm operates by directly optimizing the detection performance across the entire image,making it efficient and suitable for real-time applications.

2.2 Steps involved to detect two – wheeler helmet rider using YOLO algorithm

- **Dataset Collection and Annotation:** A dataset of images containing individuals wearing helmets is collected. Each image is annotated by marking the bounding boxes around thehelmets and labelling them as "helmet." These annotations serve as ground truth for training the YOLO model.

- **Training the YOLO Model:** The collected dataset is used to train the YOLO algorithm. Theimages and corresponding annotations are fed into the YOLO model, and the algorithm learns to recognize the visual features and patterns associated with helmets. Through an iterative process, the model adjusts its internal parameters to minimize the detection errors and improve accuracy.

- **Convolutional Neural Networks (CNN):** YOLO utilizes a CNN as its backbone network. TheCNN consists of multiple layers that perform convolution, pooling, and non-linear activation operations. These layers extract meaningful features from the input image, enabling the model to learn high-level representations of objects like helmets.

- **Anchor Boxes:** YOLO employs anchor boxes to improve detection accuracy. Anchor boxesare predefined bounding boxes of different sizes and aspect ratios that are placed on the grid cells. The YOLO model predicts the offset values and class probabilities for each anchor box, allowing it to detect objects of various sizes.

- **Non-Maximum Suppression (NMS):** After obtaining the predicted bounding boxes and their corresponding class probabilities, the YOLO algorithm applies NMS to remove redundant detections. NMS eliminates overlapping bounding boxes by selecting the one with the highest confidence score and suppressing others that have significant

overlap with it. This step ensures that each detected helmet is represented by a single bounding box.

2.3 Steps involved to detect emergency vehicle with lightbars (LED) using YOLO algorithm

Helmet detection using deep learning techniques such as YOLO (You Only Look Once) can be employed to identify motorcycle riders wearing helmets. This step helps filter out riders who arenot wearing helmets.

- **Image Extraction:** Once the helmet detection is performed, the bounding box coordinatesof the detected riders can be used to extract the corresponding image regions containingthe riders' faces and license plates. This extraction ensures that only the relevant areas are processed for license plate detection.

- **Image Pre-processing:** Pre-processing techniques may be applied to improve the quality and readability of the license plate region. Common pre-processing steps include image resizing, grayscale conversion, noise removal, contrast adjustment, and image thresholding.

- **Optical Character Recognition (OCR):** OCR is a technology that enables the recognition and extraction of text from images. In this context, OCR algorithms can be applied to the pre-processed license plate region to identify and extract the alphanumeric characters onthe license plate.

- **PyTesseract (Python Tesseract OCR):** PyTesseract is a popular Python wrapper for the Tesseract OCR engine. It provides an interface for integrating OCR capabilities into Pythonapplications. By utilizing PyTesseract, the pre-processed license plate image can be passedas input, and the corresponding text can be extracted from the license plate.

- **Text Extraction and Validation:** After applying OCR, the extracted text from the license plate can be post-processed to remove any noise or unwanted characters. Additional validation steps can be performed to ensure the accuracy and reliability of the extracted license plate information. This may include checking for the correct number of characters, format validation, or using pattern matching techniques specific to license plate formats in the target region.

- **Automated Email Generation:** Once the license plate text is successfully extracted and validated, an automated email can be generated. The email can include relevant

information such as the captured image of the rider, the extracted license plate text, and any additional details or instructions.

By combining helmet detection with OCR techniques, the proposed system can identify motorcycle riders who are not wearing helmets and then focus on extracting the license plate information from their images. This approach allows for the automated detection and capture of license plate details, enabling subsequent actions such as sending an email notification with the captured image and license plate information for further processing or enforcement purposes.

2.4 The libraries used to implement the above process:

2.4.1 Lightbars (LED) Detection on Emergency Vehicles:

- **Torch, TorchVision, and Ultralytics:** These libraries are commonly used in deep learning tasks. Torch is a popular framework for building and training deep neural networks, while TorchVision provides pre-trained models and datasets for computer vision tasks. Ultralytics is a library that extends Torch and TorchVision with additional functionalities for object detection, including Non-Maximum Suppression (NMS).

- **GPIO:** RPI.GPIO is a Python library that provides a simple interface for controlling the General Purpose Input-Output (GPIO) pins on a Raspberry Pi. It allows users to easily interact with external components, such as sensors and actuators, by reading input from and writing output to the GPIO pins, enabling various hardware-based projects and automation tasks.

- **Non-Maximum Suppression (NMS):** NMS is a post-processing technique used in object detection to remove redundant bounding box detections. After obtaining bounding box predictions for Lightbars (LED) lights, NMS selects the bounding boxes with the highest confidence scores while suppressing overlapping detections, ensuring that each detected Lightbars (LED) light is represented by a single bounding box.

- **CV2 (OpenCV):** OpenCV is a popular computer vision library that provides various image processing and computer vision algorithms. It can be used to read and manipulate images, apply filters, perform image transformations, and extract relevant features.

2.4.2 Helmet Detection on Motorcycle Riders:

- **Torch, TorchVision, and Ultralytics:** These libraries are also utilized for helmet detection. The process involves training a deep learning model using Torch and TorchVision, and Ultralytics provides additional functionalities for object detection.

- **Non-Maximum Suppression (NMS):** Similar to Lightbars (LED) light detection, NMS is employed to eliminate redundant bounding box detections in helmet detection. It selects the bounding boxes with the highest confidence scores while suppressing overlapping detections, ensuring accurate and non-duplicated helmet detections.

- **OS:** The `os` library in Python provides a set of functions for interacting with the operating system. It allows you to perform tasks such as navigating directories, manipulating files, and executing system commands. It serves as a powerful tool for managing and automating operating system-related operations within Python programs.

2.4.3 License Plate Recognition:

- **CV2 (OpenCV) and PIL (Python Imaging Library):** OpenCV and PIL are used for image manipulation and processing. They provide functions for image reading, resizing, and applying various image enhancement techniques.
- **PyTesseract (Python Tesseract OCR):** PyTesseract is a Python wrapper for Tesseract OCR (Optical Character Recognition) engine. It enables the extraction of text from images, making it suitable for recognizing and extracting characters from license plates.

2.4.4 Automated Email Sending:

- **Time and importOS:** The Time library in Python is a versatile tool that allows developers to work with time-related operations. It provides functions for tracking time, scheduling tasks, and executing them at specific intervals. This library is useful for applications that require timed events, such as scheduling automated tasks or implementing countdowns. On the other hand, the importOS module provides functions related to the operating system, allowing developers to interact with the file system, manipulate files and directories, and execute system commands. It offers

a convenient way to perform OS-specific operations and handle file-related tasks in Python.

- **SMTPlib:** SMTPlib is a crucial Python library for sending emails using the Simple Mail Transfer Protocol (SMTP). It provides a set of functions and classes that enable developers to establish connections with SMTP servers, compose email messages, and send them programmatically. With SMTPlib, developers can automate the process of sending emails, making it easier to integrate email functionality into their applications. Whether it's sending transactional emails, notifications, or reports, SMTPlib simplifies the task of email communication in Python.

- **NumPy:** It provides efficient data structures, such as arrays and matrices, along with a wide range of mathematical functions and operations. NumPy is widely used in scientific and data-intensive applications, as it offers high-performance computations and convenient tools for data manipulation. With NumPy, developers can perform tasks like matrix operations, statistical analysis, and numerical simulations. Its extensive functionality and optimized algorithms make it an essential library for numerical computing tasks in Python.

- **Email:** The Email library in Python offers a comprehensive set of classes and functions for creating, formatting, and sending email messages. It provides developers with the ability to compose emails with various content types, including plain text, HTML, and rich text. Additionally, the Email library supports the attachment of files, such as images and text files, to email messages. This feature allows developers to send emails with attachments programmatically. Whether it's sending newsletters, reports, or personalized emails, the Email library simplifies the process of email generation and management in Python applications.

By utilizing these libraries and technologies, the proposed system can detect Lightbars (LED) lightson emergency vehicles, detect helmets on motorcycle riders, recognize license plates, extract textfrom the license plates, and automatically send emails with captured images and extracted information.

3. Problem Definition

3.1 Introduction

The current traffic management systems have a number of difficulties when it comes to recognizing and validating vehicles for traffic law enforcement, effectively responding to crises, and implementing helmet usage requirements and Lightbar (LED) These difficulties lead to longer emergency response times, higher accident rates, and trouble keeping the traffic flow in a controlled manner. A pre-existing dataset for helmet detection will also be obtained, and a dataset dedicated to emergency vehicle lightbars will be generated. These datasets will be used to train the YOLO model, which will enable it to correctly identify emergency vehicles and helmets.

3.2 Problem Statement

The existing traffic management systems face challenges in effectively detecting and responding to emergency situations, enforcing helmet usage regulations, and efficiently identifying and verifying vehicles for traffic law enforcement. These challenges can lead to delayed emergency response times, increased risk of accidents, and difficulties in maintaining order on the roads. Addressing these challenges is crucial to ensure the smooth and safe functioning of traffic systems, safeguarding the well-being of both motorists and pedestrians alike. By leveraging advanced technologies, such as artificial intelligence and computer vision, there is an opportunity to overcome these challenges. Implementing intelligent systems that can accurately detect emergencies, enforce helmet usage, and efficiently identify vehicles can lead to improved response times, enhanced safety measures, and more effective traffic law enforcement.

3.3 Project Objective

- To conduct literature survey on object detection (emergency vehicles and helmet detection) techniques to be used for traffic management system.
- To develop the dataset for lightbars on the emergency vehicle and gather pre- existing dataset for helmet detection.
- To train the YOLO model using developed data set for emergency vehicle detection and using the pre-existing dataset for helmet detection.
- To design and develop a model and simulate the real-life scenarios using figurines.
- To analyze and evaluate the working model using standard metrics.

3.4 Methods and Methodology

Objective	Methodology	Description
1.	Literature Survey	Conduct survey on various deep learning and ML approaches for object detection and character recognition.
2.	Developing dataset for Emergency vehicles	Create a dataset of images containing emergency vehicle instances from different angles and lighting conditions.
3.	Training the YOLO model for vehicle detection	Uploading the created dataset to YOLO for training and cross validating the model.
4.	Training the YOLO model for helmet detection	Uploading the existing dataset from Kaggle to YOLO for training and cross validating the model.
5.	Simulation using a model	A model is created using raspberry pi to simulate the traffic system and test our results using toys.
6.	Evaluation using standard metrics	Using standard metrics like accuracy, precision and recall to analyse model performance.
7.	Scientific Report	A report highlighting all the features of the proposed system is documented as per the university template.

Table 3.1.1: Methods and Methodology

The data collection effort began with the aim of creating a traffic system for the proposed emergency vehicle detection and helmet detection model. To achieve this, reputed sources were considered, including scientific journals, articles on the internet and sources on YouTube and other platforms to gain insights into the state-of-the-art algorithms, architectures, and methodologies employed in these domains. next focus was to develop a dataset specifically for emergency vehicle detection. The diverse collection of images was crucial for training and evaluating the models accurately, enabling them to recognize emergency vehicles robustly.

We train the YOLO (You Only Look Once) model for vehicle detection. The created dataset is uploaded to the YOLO model for training and cross-validation. This process enhances the

model's ability to accurately detect and localize vehicles in various scenarios. Similarly, another YOLO model is trained specifically for helmet detection. An existing dataset from Kaggle, containing images of motorcycle riders with and without helmets, will be used for training and cross-validation. To simulate real-life traffic scenarios, a model is created using a Raspberry Pi. This model mimics a traffic system, and the trained vehicle and helmet detection models will be integrated into the simulation. This will allow for thorough testing and analysis of the models' performance using toy vehicles.

The performance of the trained models is evaluated using standard metrics such as accuracy, precision, and recall. These metrics provides quantitative measures of the models' effectiveness in detecting emergency vehicles, helmets, and license plates. The evaluation will help identify areas for improvement and optimization, enabling the refinement of the models' performance. Finally, a scientific report is prepared, documenting all the features of the proposed system. The report follows the university's designated template and provide a comprehensive overview of the project's objectives, methodologies, and outcomes.

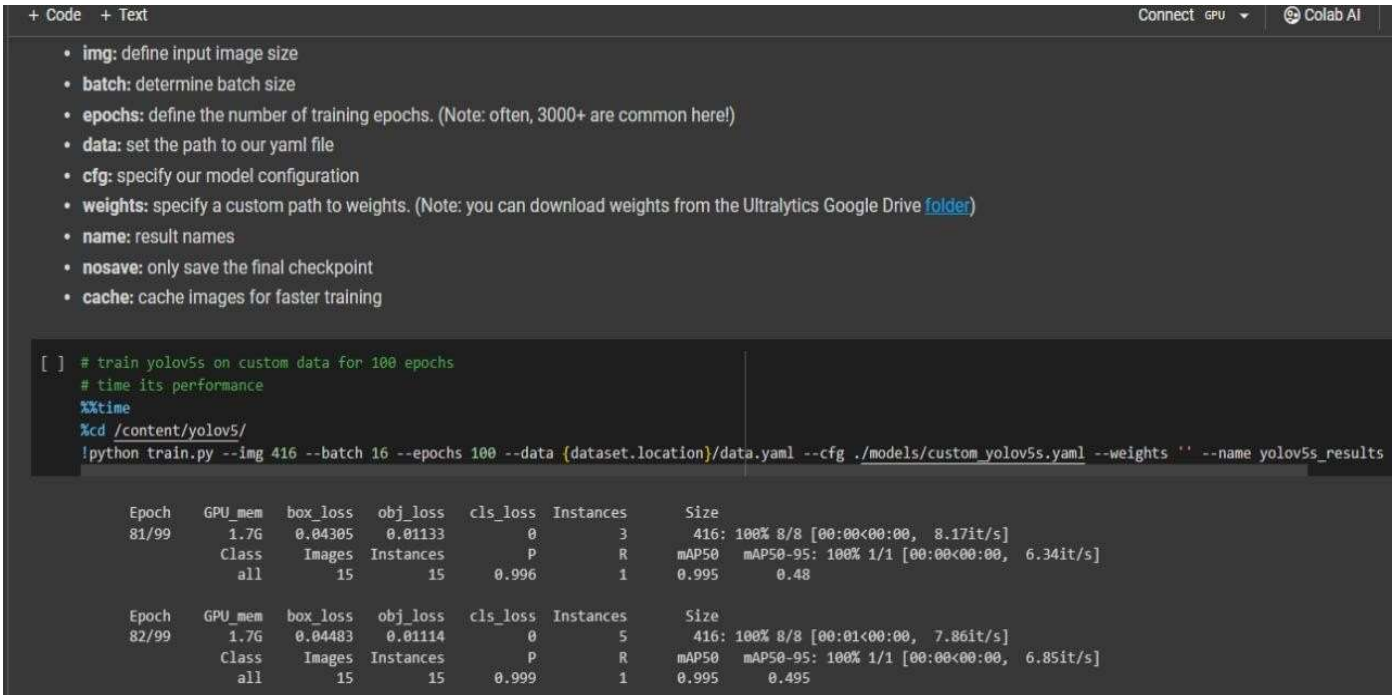


Figure 3.4.1 – Training on the custom data

Block Diagrams:

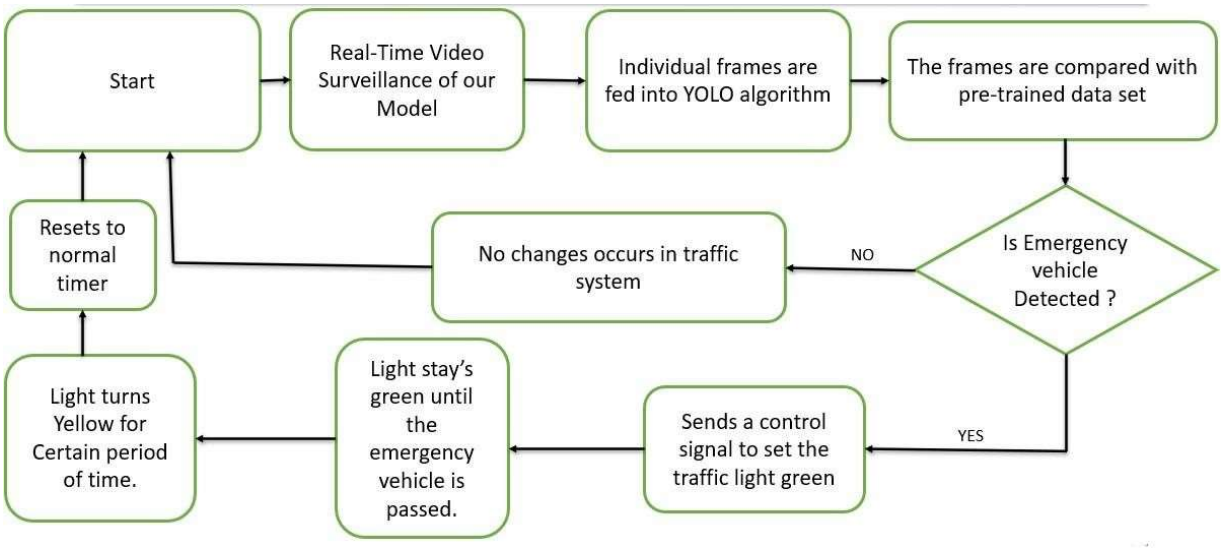


Figure 3.4.2 Block Diagram for Emergency Vehicle

The block diagram outlines a process for real-time video surveillance to detect emergency vehicles and adjust traffic lights accordingly. The process begins with real-time video surveillance of a model, where individual frames are captured and fed into the YOLO (You Only Look Once) algorithm. YOLO is a state-of-the-art object detection system used in computer vision. The individual frames processed by the YOLO algorithm are then compared with a pre-trained dataset. This dataset contains information that helps the system identify whether an emergency vehicle is present in the frame.

If the system detects an emergency vehicle in the frame, it triggers a control signal. If no emergency vehicle is detected, the traffic system remains unchanged and the process resets to the normal timer. However, if an emergency vehicle is detected, a control signal is sent to the traffic light system. This signal changes the traffic light to green, allowing the emergency vehicle to pass quickly and safely. The light stays green until the emergency vehicle has passed.

After the emergency vehicle has passed, the light turns yellow for a certain period before returning to its normal operation. This system could significantly improve response times for emergency vehicles by providing them with a clear path through traffic. Furthermore, the system can be integrated with other smart city infrastructure for more comprehensive urban management. For instance, the data collected can be used to analyze traffic patterns, plan road infrastructure, and develop strategies for reducing congestion. This system is a significant advancement in traffic management and emergency response. By using real-time video surveillance and the YOLO algorithm, it ensures that emergency vehicles can navigate through traffic more efficiently, potentially saving lives in the process.

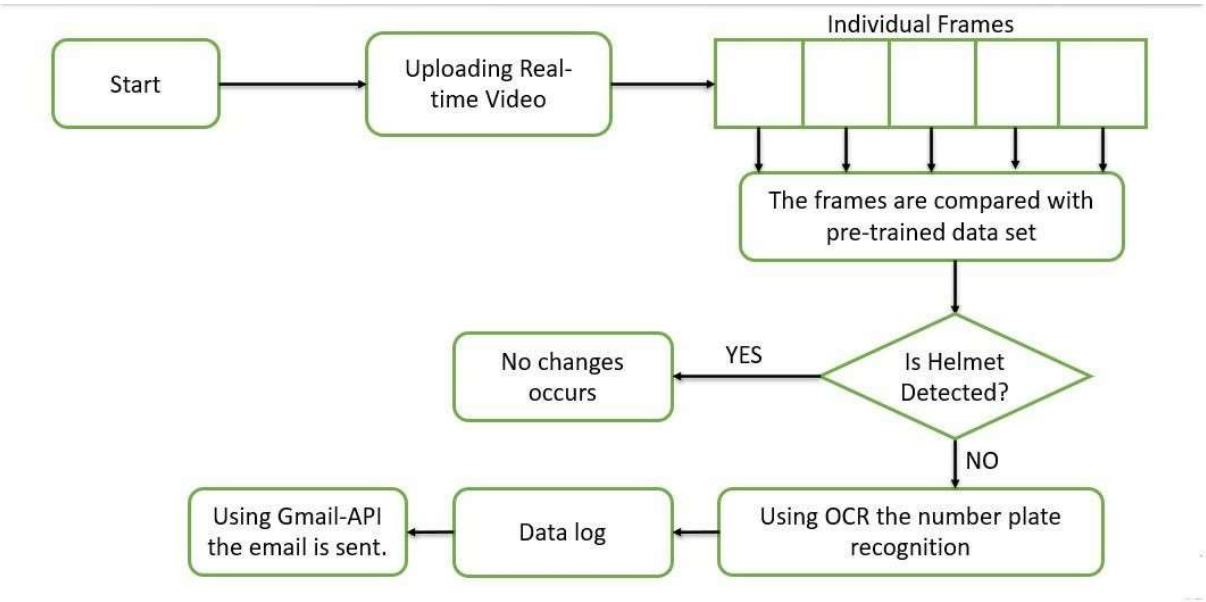


Figure 3.4.3 Block Diagram for Helmet Detection

The block diagram you provided outlines a process for real-time video surveillance to detect whether individuals are wearing helmets and for number plate recognition. The process begins with uploading real-time video, which is then broken down into individual frames. These frames are compared with a pre-trained dataset to identify if there are any helmets present in the frames.

If helmets are detected, data is logged and an email is sent using Gmail-API to notify relevant parties of the outcome. If helmets are not detected, OCR technology is employed to recognize the number plates of vehicles involved. This comprehensive approach ensures that safety protocols are adhered to and allows for efficient tracking and notification mechanisms in case of any violations.

This system is a significant advancement in safety monitoring and traffic management. By using real-time video surveillance, pre-trained datasets for helmet detection, OCR for number plate recognition, and Gmail-API for sending emails, it ensures that safety regulations are being followed. Furthermore, the system can be integrated with other smart city infrastructure for more comprehensive urban management. For instance, the data collected can be used to analyze traffic patterns, plan road infrastructure, and develop strategies for reducing congestion. This system is a significant advancement in traffic management and safety monitoring. By using real-time video surveillance, pre-trained datasets, OCR, and Gmail-API, it ensures that safety regulations can be monitored more efficiently, potentially improving safety in the process.

4. Problem Solving

4.1 Introduction

Building upon the methodologies discussed in the preceding chapter, this section delves into the step-by-step development process of the prototype, adhering to the methodologies outlined to achieve each predefined objective. The chapter meticulously outlines the deliverables achieved at the culmination of each developmental stage. It expounds upon the analysis and design models acquired, including the block diagrams, instrumental for implementation. The implemented prototype system is elucidated, incorporating detailed code listings. The testing phases are transparently elucidated, encompassing the test suite, objectives, and outcomes. Ultimately, a comprehensive analysis of the system's accuracy and scalability is presented.

4.2 Problem Solving Approach

4.2.1 Collecting and annotating pictures for training

Collecting and annotating pictures for training is a crucial step in developing machine learning models, specifically for tasks like recognizing and classifying pictures using YOLO (You Only Look Once) and OCR (Optical Character Recognition) techniques. This process involves several important considerations:

It is essential to clearly define the objectives of the machine learning model. This includes specifying the desired outcomes and the specific criteria for recognizing and classifying the pictures. By establishing clear objectives, developers can focus their efforts on collecting and annotating images that align with the intended use case.

Creating a diverse and representative dataset becomes crucial. In this context, the dataset needs to contain a wide range of images that are relevant to the task at hand, specifically related to Lightbars (LEDs) of emergency vehicles. Gathering a diverse set of images ensures that the model can handle various scenarios and generalizes well to unseen data. It is also important to ensure that the dataset is balanced, meaning it contains a reasonable distribution of different classes or categories of Lightbars (LEDs). This helps prevent bias and ensures that the model is trained on a representative sample of data.

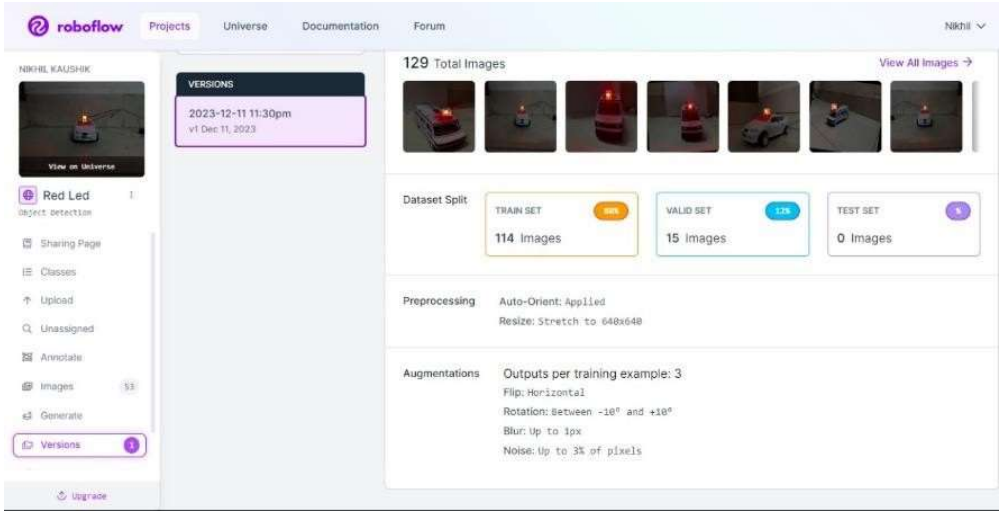


Figure 4.2.1- Annotation of the dataset

4.2.2 Data Pre-processing and Splitting the Dataset

Resize, crop, and normalize the images of the Lightbars (LED) using roboflow platform to ensure uniformity in the dataset and Converting annotations into a format suitable for

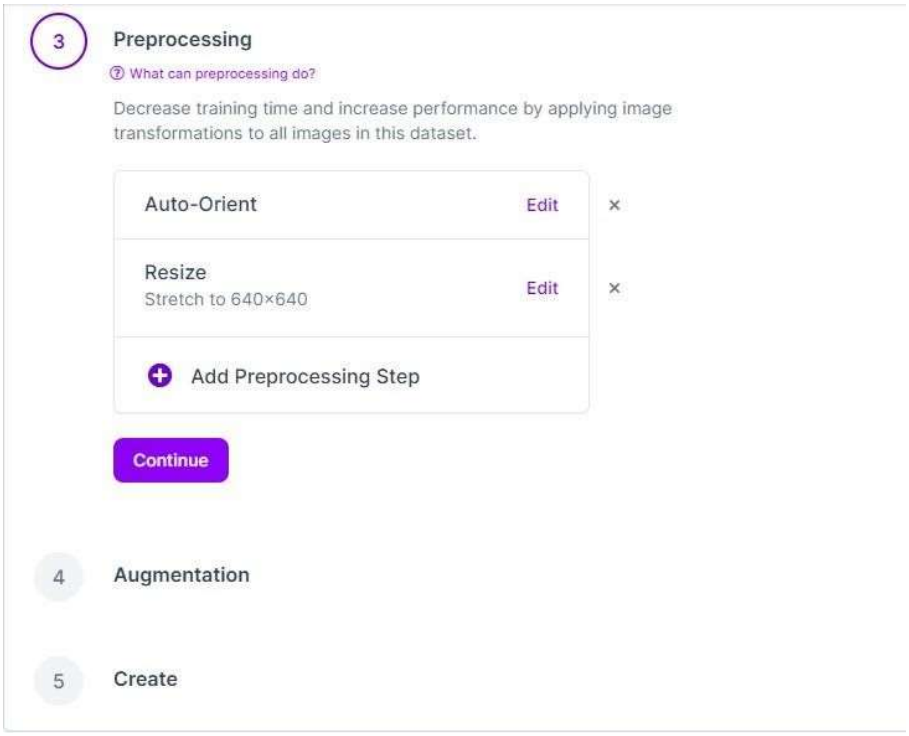


Fig 4.2.2- Preprocessing the dataset

4.2.3 Training and Generation of Weight File

Setting up the environment for training our YOLO model. This involves installing the necessary libraries, frameworks which includes **Torch, Torch Vision, and Ultralytics etc**. Using a deep learning framework TensorFlow, PyTorch to train our model on the annotated dataset. And adjusting the hyperparameters based on the performance on the validation set.

After training the YOLO model, the weights of the model are typically saved in a file for future use or deployment. The exact process of saving and loading weights depends on the framework. For helmet detection the dataset has been collected from KAGGLE and trained.

4.2.4 Implementation of ambulance detection

4.2.4.1 Setting up the Raspberry Pi and GPIO pins

Set up the Raspberry Pi and connect the GPIO pins to the traffic lights Identify the GPIO pins that correspond to the red, yellow, and green LEDs of each traffic light. These pins will be used to control the lights. Connect the GPIO pins to the appropriate LEDs of the traffic lights.

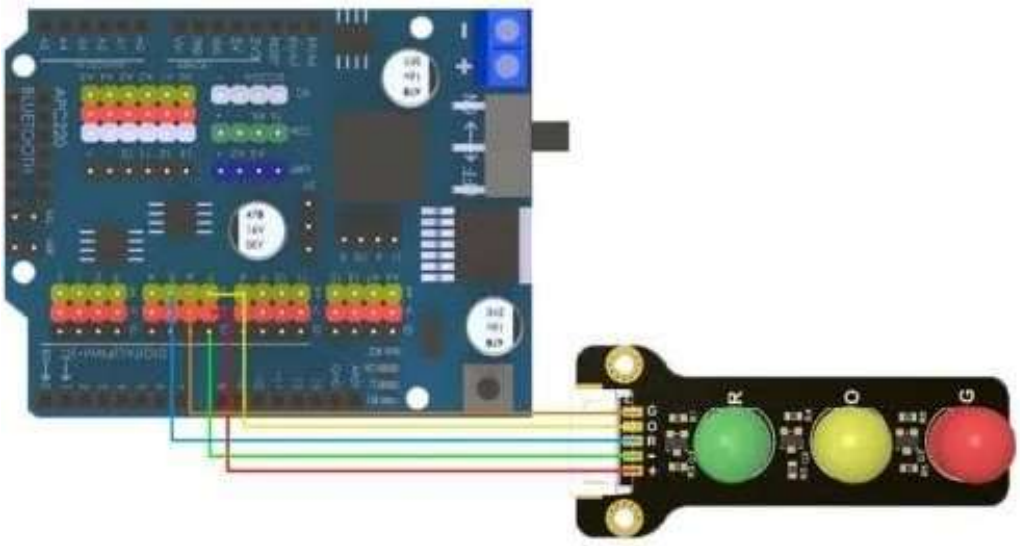


Figure 4.2.4.1 – Traffic light module connection with Raspberry Pi module

4.2.4.2 Installing the necessary libraries

- **OpenCV:** Installing the OpenCV library on our Raspberry Pi. we can use the following command to install it: **pip install OpenCV-python.**
- **NumPy:** Installing the NumPy library, which is a dependency for OpenCV. Using the command: **pip install NumPy.**
- **RPi.GPIO:** Installing the RPi.GPIO library for controlling the GPIO pins. We can use the following command to install and Run: **pip install RPi.GPIO.**
- **Torch:** Installing PyTorch, which is required for the YOLOv5 model. Depending on our Raspberry Pi model, we may need to build PyTorch from source or use a prebuilt version compatible with our architecture.

4.2.4.3 Downloading the YOLOv5 model

Obtaining the YOLOv5 model weights file (yolov5s.pt) from the official YOLOv5 repository on GitHub. Save the weights file in a directory accessible to our Raspberry Pi.

4.2.4.4 Running the code on the Raspberry Pi

- Transfer the **ambulance_detect.py** file to our Raspberry Pi.
- Open a terminal or SSH session on the Pi.
- Navigate to the directory where we saved the **ambulance_detect.py** file.
- Run the code using the Python interpreter: python ambulance_detect.py.
- Observe the terminal output, which will display the current state of the traffic lights and any detected ambulance activity.

4.2.4.5 Monitoring the camera feed and detecting red Lightbars (LED)

Ensuring that a camera is connected to the Raspberry Pi and recognized by the system. The code will continuously access the camera feed and process each frame. If the code detects a red Lightbars (LED) (assumed to be an ambulance light) with sufficient confidence, it will set the red detected variable to indicate the presence of an ambulance.

4.2.4.6 Traffic light control and prioritizing ambulance passage

The trafficlights () function manages the traffic lights based on the detected ambulance. When an ambulance is detected (indicated by red detected being True), the traffic lights will be adjusted to prioritize its passage. The code will interrupt the regular traffic flow and provide a green signal to the side where the ambulance is detected. Once the ambulance has passed, the code will resume the regular cyclic traffic light sequence.

4.2.4.7 Using Clean-up function

The program includes a cleanup () function that is registered to be called when the program exits. This function ensures proper cleanup of the GPIO pins by setting them to their initial state. It is important to register this function to prevent leaving the GPIO pins in an undefined state when the program terminates.

Important Code Snippets for Vehicle Detection:

```
# Load the YOLOv5 model
yolov5_weight_file = 'your weights path'
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = attempt_load(yolov5_weight_file, map_location=device)
model.conf = 0.45  # Set the confidence threshold

vehicle_detected = 0

# Main function for vehicle detection
def main_function():
    def detected(frame):
        global vehicle_detected

        # Preprocess the frame for inference
        frame = cv2.resize(frame, (800, 480))
        img = frame[:, :, ::-1]  # BGR to RGB
        img = img / 255.0  # Normalize to [0, 1]
        img = np.transpose(img, (2, 0, 1))  # HWC to CHW
        img = torch.from_numpy(img).float().to(device).unsqueeze(0)  # Add batch
dimension

        with torch.no_grad():
            results = model(img)
```

```
        results = non_max_suppression(results, 0.45, 0.45)

        # Check if results are valid
        if results[0] is None or len(results[0]) == 0:
            print("No results returned by model.")
            return frame

        # Get the results of vehicle detection
        result = results[0].cpu().numpy()

        # Draw bounding boxes around detected vehicles
        for *xyxy, conf, cls in result:
            if int(cls) == 0:  # Assuming vehicle class index is 0
                xmin, ymin, xmax, ymax = [int(x) for x in xyxy]

                # Draw bounding box
                cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 2)

                # Display confidence score and label
                label = f'siren: {conf:.2f}'
                cv2.putText(frame, label, (xmin, ymin - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

                # Set value of vehicle_detected based on confidence score
                if conf >= 0.45:
                    vehicle_detected = 1
                else:
                    vehicle_detected = 0

        return frame

    # Open webcam
    cap = cv2.VideoCapture(0)

    while True:
        # Read frame from webcam
        ret, frame = cap.read()

        # Perform vehicle detection
        output_frame = detected(frame)

        # Display the output frame
        cv2.imshow('Vehicle Detection', output_frame)

        # Check for key press to exit
```

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

        # Print the value of vehicle_detected
        print(f"vehicle_detected: {vehicle_detected}")

    # Release the webcam and close windows
    cap.release()
    cv2.destroyAllWindows()
```

This Python code implements a real-time webcam application for detecting and visualizing red LED lights using a quantized YOLOv5 model. The code uses the Ultralytics YOLOv5 repository to load a pre-trained YOLOv5 model from a specified path. The model is then quantized to reduce its precision using dynamic quantization.

4.4 Implementation of Helmet detection

4.4.1 Importing Libraries

Installing and importing the necessary libraries, including OpenCV (cv2), Torch, and other required modules.

4.4.2 Model and Weight File Path

We will define the paths to the helmet classifier weight file (helmet classifier weight). the confidence threshold for object detection is set using **conf_set**. The desired frame size is specified using **frame_size**.

4.4.3 Image Classification Model

The YOLOv5 model is loaded onto the device. the code loads a separate image classification model (model2) used for classifying images as having helmets or not. and the model is set to evaluation mode.

4.4.4 Helmet Classification Function

The img_classify() function takes an image frame as input and performs helmet classification. And the image is pre-processed using the defined transformation pipeline, the pre-processed image is passed through the image classification model (model2) to

obtain a prediction and the predicted class index and confidence scores are processed to determine if a helmet is present or not.

4.4.5 Object Detection Function

The object_detection() function takes an image frame as input and performs object detection using the YOLOv5 model. And the image frame is converted to a Torch tensor and pre-processed for inference. Object detection is performed on the pre- processed image using the YOLOv5 model.

Non-maximum suppression is applied to filter out redundant detections based on confidence scores. And the function iterates over the detections, extracting the bounding box coordinates, confidence, and class label.

The detected objects are categorized into "head," "number," or "rider" based on the class label and the bounding boxes and labels are drawn on the frame using OpenCV and the function returns the modified frame and a list of detection results.

4.4.6 Email Sending Function

The send_email () function is defined to send an email with attachments. It takes the subject, body, recipient email address, attachment paths, sender email address, sender password, SMTP server, and SMTP port as inputs. And the function creates a MIME message and attaches the specified files and it establishes a connection with the SMTP server, logs in using the sender's credentials, and sends the email.

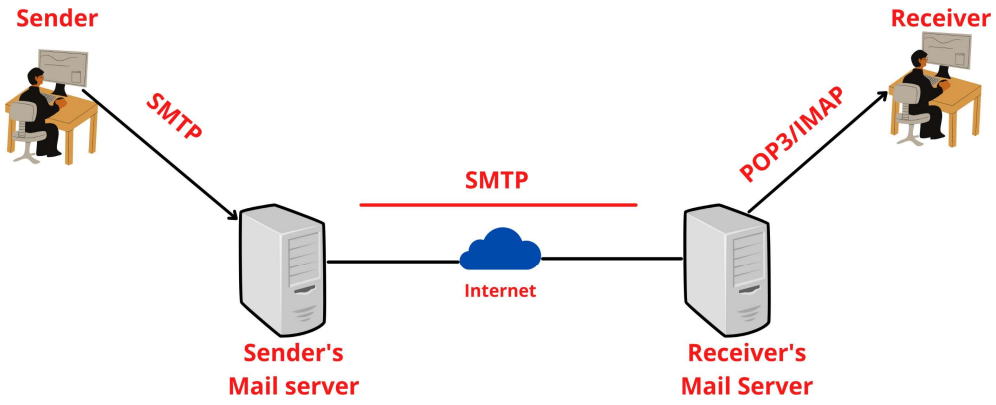


Figure 4.4.6 – Simple mail transfer protocol

4.4.7 Image Processing and OCR Function

The process_image_and_ocr() function is defined to process images for optical character recognition (OCR).

this function resizes the image, performs pre-processing steps (e.g., denoising, blurring, thresholding), and applies OCR using Tesseract. the resulting license plate text is displayed and saved to a file.

4.4.8 Utility Function

The inside_box () function checks if a smaller bounding box is entirely inside a larger bounding box.

4.4.9 Video and Image Processing Loop

This sets up video source and output parameters if a video file is used. It initializes the video writer for saving the processed video (if enabled) and the code enters a loop to read frames from the video source.

Each frame is resized, and object detection is performed using the object_detection() function and the detected objects are categorized, and the highest confidence rider is determined.

If a rider is detected, the code checks for the presence of a head within the rider's bounding box. If a head is present, helmet classification is performed using the img_classify() function. And the resulting frame is displayed, and video output is saved if enabled, this loop continues until the video ends or an interruption occurs.

Important Code Snippets for helmet detection:

```
input_path = 'path_to_input_video_or_image'  # Input video or image file
output_path = 'path_to_output_video'  # Output video file path
save_video = False  # Flag to save output video
show_video = True  # Flag to show output video
save_img = False  # Flag to save output image

# Open video capture
cap = cv2.VideoCapture(input_path)
if save_video:
```

```
    fourcc = cv2.VideoWriter_fourcc(*'MJPG')
    out = cv2.VideoWriter(output_path, fourcc, 20.0, frame_size)

# Process video frames
while cap.isOpened():
    ret, frame = cap.read()  # Read a frame
    if ret:
        orifinal_frame = frame.copy()  # Copy the original frame

        # Perform object detection
        frame, detection_result = object_detection(frame)

        # Separate detections by class
        rider_list, head_list, number_list = [], [], []
        for result in detection_result:
            x1, y1, x2, y2, conf, clas = result
            if clas == 0:
                rider_list.append(result)
            elif clas == 1:
                head_list.append(result)
            elif clas == 2:
                number_list.append(result)

        # Find the highest confidence rider
        highest_confidence_rider = max(rider_list, key=lambda x: x[4]) if
rider_list else None

        if highest_confidence_rider:
            x1r, y1r, x2r, y2r, cnfr, clasr = highest_confidence_rider
            for hd in head_list:
                x1h, y1h, x2h, y2h, cnfh, clash = hd
                if inside_box([x1r, y1r, x2r, y2r], [x1h, y1h, x2h, y2h]):  #
Check if head is inside rider bbox
                    try:
                        head_img = orifinal_frame[y1h:y2h, x1h:x2h]
                        helmet_present = img_classify(head_img)
                    except:
                        helmet_present[0] = None

                    if helmet_present[0] == True:  # Helmet present
                        frame = cv2.rectangle(frame, (x1h, y1h), (x2h, y2h), (0,
255, 0), 1)
                        frame = cv2.putText(frame, f'{round(helmet_present[1],
1)}', (x1h, y1h + 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1,
cv2.LINE_AA)
```

```
                    elif helmet_present[0] == None:  # Poor prediction
                        frame = cv2.rectangle(frame, (x1h, y1h), (x2h, y2h), (0,
255, 255), 1)
                        frame = cv2.putText(frame, f'{round(helmet_present[1],
1)}', (x1h, y1h), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    elif helmet_present[0] == False:  # Helmet absent
                        frame = cv2.rectangle(frame, (x1h, y1h), (x2h, y2h), (0,
0, 255), 1)
                        frame = cv2.putText(frame, f'{round(helmet_present[1],
1)}', (x1h, y1h + 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1,
cv2.LINE_AA)
                        try:
                            cv2.imwrite(f'riders_pictures/{time_stamp}.jpg',
frame[y1r:y2r, x1r:x2r])
                        except:
                            print('could not save rider')

                        # Find the highest confidence number plate
                        highest_confidence_number = max(number_list, key=lambda
x: x[4]) if number_list else None

                        if highest_confidence_number:
                            x1_num, y1_num, x2_num, y2_num, conf_num, clas_num =
highest_confidence_number
                            if inside_box([x1r, y1r, x2r, y2r], [x1_num, y1_num,
x2_num, y2_num]) and conf_num > 0.4:
                                try:
                                    num_img = orifinal_frame[y1_num:y2_num,
x1_num:x2_num]
                                    cv2.imwrite(f'number_plates/{time_stamp}_{con
f_num}.jpg', num_img)
                                    process_image_and_ocr(num_img, time_stamp,
conf_num)

                                    # Collect file paths for attachments
                                    rider_image_path =
f'riders_pictures/{time_stamp}.jpg'
                                    number_plate_image_path =
f'number_plates/{time_stamp}_{conf_num}.jpg'
                                    number_text_path =
f'numbers/{time_stamp}.txt'
                                    attachment_paths = [rider_image_path,
number_plate_image_path, number_text_path]

                                    # Specify email details
```

```
                                    subject = "Helmet Detection Results"
                                    body = "Please find the attached images and
text files for helmet detection results."
                                    to_email = "reciver mail"
                                    sender_email = "your mail"
                                    sender_password = "your password"
                                    smtp_server = "smtp.gmail.com"
                                    smtp_port = 587

                                    try:
                                        send_email(subject, body, to_email,
attachment_paths, sender_email, sender_password, smtp_server, smtp_port)
                                    except Exception as e:
                                        print(f"Error sending email: {e}")

                                except:
                                    print('could not save number plate')

        if save_video:  # Save video
            out.write(frame)
        if save_img:  # Save image
            cv2.imwrite('saved_frame.jpg', frame)
        if show_video:  # Show video
            frame = cv2.resize(frame, (850, 480))  # Resize to fit the screen
            cv2.imshow('Frame', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):  # Break the loop if 'q' is pressed
            break

    else:
        break

cap.release()  # Release video capture
cv2.destroyAllWindows()  # Close all OpenCV windows
print('Execution completed')
```

5. Results

5.1 Introduction

After the integration of hardware and software, the testing is carried out and the results are obtained. In this chapter, the obtained results are tabulated and plotted.

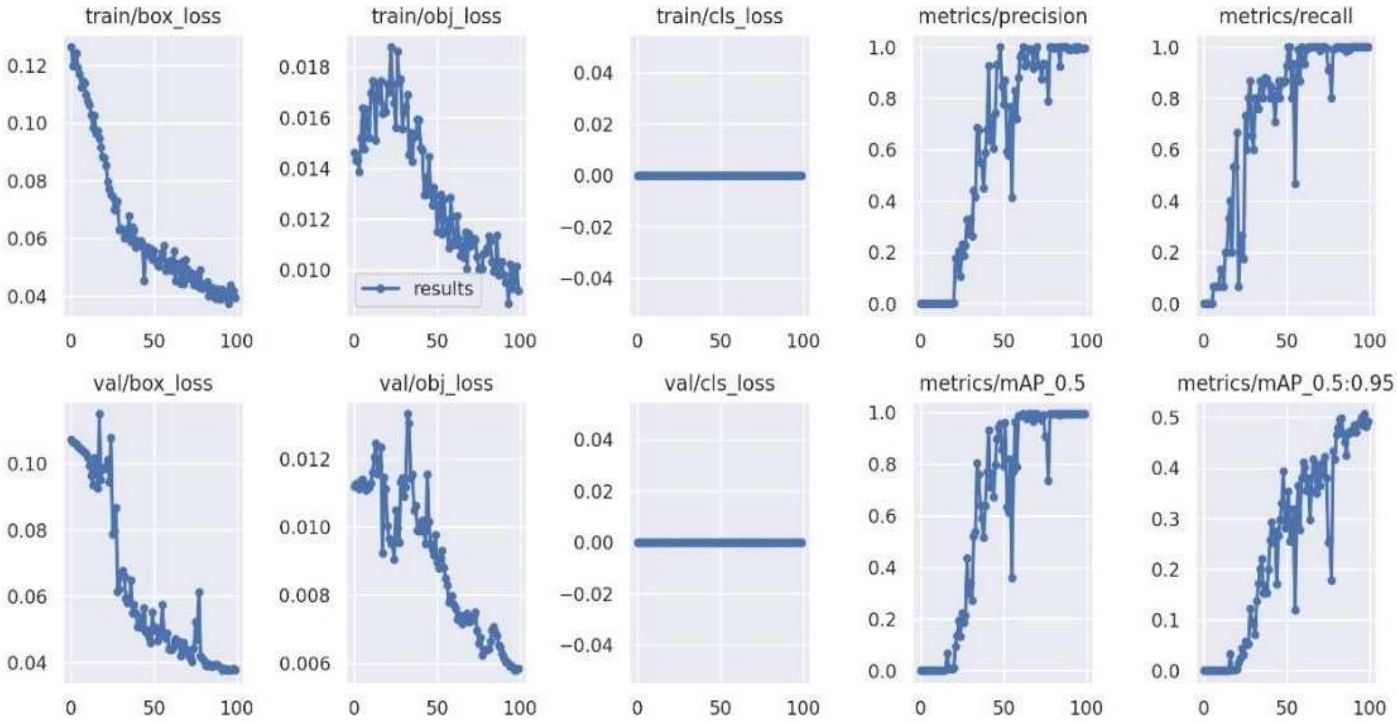
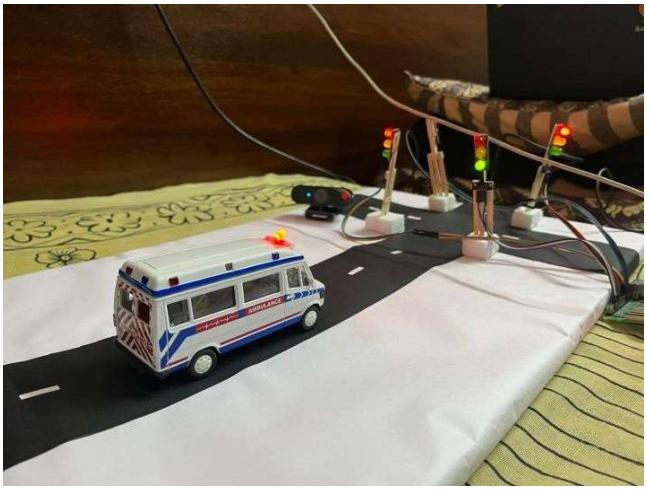


Figure 5.1 – Training loss and mean average precision

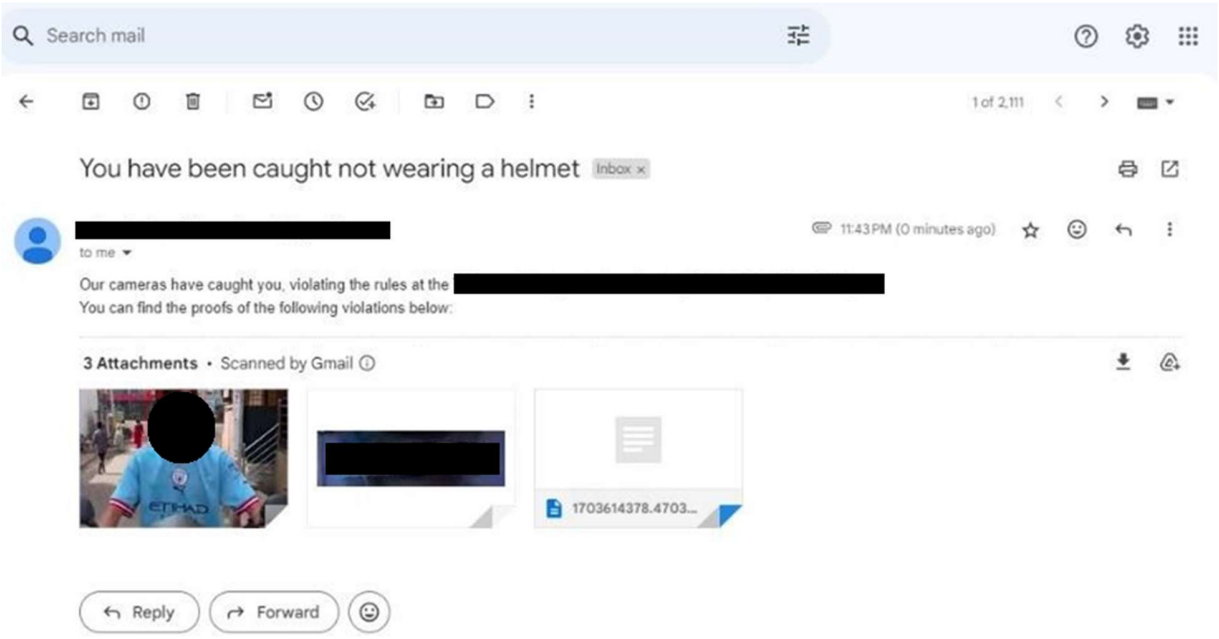
5.2 Traffic Management System for Emergency vehicle detection and helmet detection

The image below shows the model we developed to mimic the real-world scenario we've used raspberry pi to control the traffic module. It has a synchronous timer system to all the four traffic modules like in real world, if it detects an emergency vehicle it turns the corresponding signal green and turns all the other signal red.



Traffic module indication

The captured images and the text extracted from the license plate is sent to the rider through an automated email system



Email

6. Conclusions and Suggestions for Future Work

6.1 Introduction

In this project, we have presented a comprehensive framework for real-time traffic management, focusing on the detection and recognition of emergency vehicles, automated helmet detection, and license plate recognition. The system utilizes the You Only Look Once (YOLOV5) object detection algorithm to achieve these functionalities.

6.2 Conclusion

In conclusion, this project has presented a robust traffic management system utilizing computer vision techniques.

The Lightbars (LED) detection component of the system allows for the efficient identification of emergency vehicles equipped with Lightbars (LED)s. This enables timely recognition of emergency situations, leading to faster response times and reduced traffic disruptions. The system also incorporates signal control, automatically adjusting traffic signals to prioritize the passage of emergency vehicles. This feature improves traffic flow and minimizes delays, contributing to more efficient emergency response.

Additionally, the system includes automated helmet detection, which enhances road safety by identifying motorcyclists and detecting whether they are wearing helmets. This functionality aids in enforcing helmet regulations and enables authorities to take appropriate actions in real-time to ensure rider safety.

By integrating these functionalities into a single system, we have demonstrated the potential of computer vision techniques to enhance traffic management and safety. The real-time capabilities of the YOLOV5 algorithm enable timely detection and response to emergency situations, efficient traffic flow, and enforcement of traffic regulations.

6.3 Suggestion for future work

While this project provides a solid foundation for traffic management and safety, there are several areas that can be explored for future enhancements and improvements. Some suggestions for future work include:

- **Real-time Traffic Analytics:** Integrate advanced analytics capabilities into the system to gather and analyse real-time traffic data. This can provide insights into traffic patterns, congestion hotspots, and optimize traffic signal timings for improved traffic flow.

- **Intelligent Routing:** Develop an intelligent routing system that considers real-time traffic conditions, emergency vehicle locations, and road network data to generate optimal routes for emergency vehicles. This can further reduce response times and improve efficiency.

- **Multi-camera Integration:** Extend the system to support integration with multiple cameras placed at different locations. This would allow for broader coverage and better monitoring of traffic violations and emergency situations.

- **Integration with Emergency Services:** Establish seamless integration with emergency service providers, such as ambulance services and police departments. This would enable direct communication and coordination between the system and emergency responders for more effective incident management.

- **Mobile Application:** Develop a mobile application that allows users to report traffic violations, emergency situations, or road hazards. This crowdsourced data can be integrated into the system for improved real-time monitoring and response.

- **Continuous Model Training:** Implement a mechanism to continuously update and retrain the object detection models using new data. This would ensure that the system remains accurate and effective over time, accounting for changes in traffic patterns and vehicle appearances.

- **Collaborative Traffic Management:** Enable collaboration between different traffic management systems deployed in various regions or cities. This would facilitate information sharing, coordinated traffic management, and seamless travel across different jurisdictions.

By exploring these suggestions for future work, the system can be further enhanced to provide more advanced and comprehensive traffic management and safety features, contributing to safer and more efficient roadways.

References

[1] Vision-based vehicle detection and counting system using deep learning in highway scenes (2019) Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, and Xu Yun.

[2] Ambulance detection using image processing and neural networks (2022) K Agrawal, M K Nigam, S Bhattacharya, Sumathi G.

[3] A New Hybrid Architecture for Real-Time Detection of Emergency Vehicles (2021) Eshwar Prithvi Jonnadula and Pabitra Mohan Khilar.

[4] Helmet Detection on Motorcyclists Using Image Descriptors and Classifiers (2014) Romuere Silva, Kelson Aires and Rodrigo Veras.

[5] Lu presented a paper titled "Safety Helmet Detection for ATM Surveillance System Using Modified Hough Transform" at the IEEE 37th Annual International Carnahan Conference on Security Technology in 2003, with pages 364–369.

[6] C.-C. Chiu, M.-Y. Ku, and H.-T. Chen, “Motorcycle detection and tracking system with occlusion segmentation,” in WIAMIS ’07, USA, 2007.

[7] J. Chiverton, “Helmet presence classification with motorcycle detection and tracking,”

IET, vol. 6, no. 3, pp. 259–269, 2022.

[8] Deep Learning-Based Safety Helmet Detection in Engineering Management Based on Convolutional Neural Networks (2020) Yange Li, Han Wei, Zheng Han, Jianling Huang and Weidong Wang.

[9] Laganière R 2011 OpenCV 2 Computer Vision Application Programming Cookbook Pub.

[10] Times of India:

- https://www.timesnownews.com/mirror-now/in-focus/bengaluru-ambulance- delays-leave-citizens-fuming-in-the-city-article-92187682
- https://timesofindia.indiatimes.com/city/raipur/choking-life-12-on-way-to- hospitals-die-in-traffic-jams/articleshow/54910538.cms

[11] Radhe Disaster and Education Foundation: https://radhee.com/advocacy/morth-road-safety/

[12] Ministry of Road Transport and Highway: https://morth.nic.in/road-accident-in-india

[13] Bengaluru District Police:
https://bangaloreruralpolice.karnataka.gov.in/page/Traffic/Road+Accidents/en

[14] Abdoos M., Mozayani N and Bazzan A.L.C., "Traffic Light Control in Non-Stationary Environments based on Multi agent Q-learning", in Proc. IEEE Conference on Intelligent Transportation Systems, pp.580- 1585, 2021.

[15] Suresh Sharma, Alok Pithora, Gaurav Gupta, Mohit Goel, Mohit Sinha, "Traffic Light Priority Control for Emergency Vehicle Using RFID", International Journal of Innovations in Engineering and Technology, 2022.

[16] Legon-Okponglo, "Design and development of microcontroller-based traffic system using image processing techniques". University of Ghana,published in ICAST, 2012 IEEE 4th International Conference

[17] "Automatic Helmet Detection in Real-Time and Surveillance Video" by Shubham Kumar, Nisha Neware, Atul Jain, Debabrata Swain & Puran Singh

[18] "Deep Learning-Based Safety Helmet Detection in Engineering Management Based on Convolutional Neural Networks" by Yange Li, Han Wei, Zheng Han, Jianling Huang, and Weidong Wang

[19] "Real-Time Helmet Violation Detection Using YOLOv5 and Ensemble Learning"

[20] "Improved OCR based automatic vehicle number plate recognition using features trained neural network"

[21] "Automatic Number Plate Recognition" by Vanshika Rai, Deepali Kamthania

[22] "Sending mail with the Mail API - Google Cloud

[23] "Automatic Helmet Detection in Real-Time and Surveillance Video" by Shubham Kumar, Nisha Newari, Atul Jain, Debabrata Swain & Puran Singh

[24] "A Real-Time Safety Helmet Wearing Detection Approach Based on CSYOLOv3" by Haikuan Wang, Zhaoyan Hu, Yuanjun Guo, Zhile Yang, Feixiang Zhou, and Peng Xu

[25] "Deep Learning-Based Safety Helmet Detection in Engineering Management Based on Convolutional Neural Networks" by Yange Li, Han Wei, Zheng Han, Jianling Huang, and Weidong Wang

[26] "Automatic Number Plate Recognition" discusses the development of an automatic number plate detection and recognition system for Indian vehicles

[27] "Ambulance Detection System" discusses the development of a computer vision system to robustly detect ambulances observed from a static camera

[28] A. Otlora, D. Osorio, and N. Moreno, “Methods for extraction of features and discrimination of emergency sirens,” pp. 1525–1532, 2017.

[29] F. Meucci, L. Pierucci, E. Del Re, L. Lastrucci, and P. Desii, “A realtime siren detector to improve safety of guide in traffic environment,” 2008 16th European Signal Processing Conference, pp. 1–5, 2008.

[30] “MOTORCYCLE SAFETY HELMET DETECTION FOR THE TWO WHEELER MOTORCYCLIST” presents a comprehensive review of the current state-of-the-art in computer vision-based helmet detection systems for two-wheelers

[31] Koller, D., Weber, J., Haung, T., Malik, J., Ogasawara, G., Rao, B., Russel, Towards robust automatic traffic scene analysis in realtime.