VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

A Project Report

on

Roadwatch: Traffic violation detection and monitoring using Image Processing

Submitted in partial fulfillment of the requirements for the final year degree in
Bachelor of Engineering in Computer Science and Engineering
of Visvesvaraya Technological University, Belagavi

Submitted by

Abhinav Kumar	1RN21CS007
Amith Sagar L	1RN21CS019
Arjun P Chander	1RN21CS033
Chirag N Sundar	1RN21CS047

Under the Guidance of
Mr.Sanjay P Kalas
Assistant Professor
Dept. of CSE

Department of Computer Science and Engineering
RNS Institute of Technology
Affiliated to VTU, Recognized by GOK, Approved by AICTE,New Delhi
NACC 'A + Grade' Accredited, NBA Accredited (UG-CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098
Ph:(080)28611880, 28611881 URL:www.rnsit.ac.in

2024-2025

RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Recognized by GOK, Approved by AICTE, New Delhi
NACC 'A + Grade' Accredited, NBA Accredited (UG-CSE,ECE,ISE,EIE and EEE)
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098
Ph:(080)28611880,28611881 URL:www.rnsit.ac.in

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

Certified that the Project work entitled **Roadwatch: Traffic violation detection and monitoring using Image Processing** has been successfully carried out at **RNSIT** by **Amith Sagar L** bearing **1RN21CS019**, **Chirag N. Sundar** bearing **1RN21CS047**, **Abhinav Kumar** bearing **1RN21CS007** ,and **Arjun P Chander** bearing **1RN21CS033** bonafide students of **RNS Institute of Technology** in partial fulfillment of the requirements of final year degree in **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi** during academic year **2024-2025**. The Project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Mr. Sanjay P Kalas	**Dr. Kavitha C**	**Dr. Ramesh Babu H S**
Assistant Professor	Dean and Head	Principal
Dept. of CSE, RNSIT	Dept. of CSE , RNSIT	RNSIT

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

Acknowledgement

I extend my profound thanks to the **Management of RNS Institute of Technology** for fostering an environment that promotes innovation and academic excellence.

I want to express my gratitude to our beloved Director, **Dr. M K Venkatesha**, and Principal,

Dr. Ramesh Babu H S , for their constant encouragement and insightful support.Their guidance has been pivotal in keeping me motivated throughout this endeavour.

My hearfelt appreciation goes to **Dr. Kavitha C**, Professor and HoD of Computer Science and Engineering, for her vital advice and constructive feedback, which significantly contributed to shaping this project.

I also thank, **Prof. Anupama**, Assistant Professor, Project Coordinator, for her continuous monitoring and ensuring the process was deployed as per schedule.

I am deeply grateful to project guide, **Mr.Sanjay P Kalas**, Assistant Professor, for his unwavering support, guidance, and valuable suggestions throughout the duration of this project. Lastly my thanks go to all the teaching and non-teaching staff members of the Computer Science and Engineering Department for their encouragement, cooperation and support have been invaluable during this journey.

Warm Regards,

Abhinav Kumar (1RN21CS007)

Amith Sagar L (1RN21CS019)

Arjun P Chander (1RN21CS033)

Chirag N Sundar (1RN21CS047)

Abstract

This project envisions the integration of cutting-edge technologies into a comprehensive smart helmet and real-time traffic monitoring ecosystem aimed at transforming urban traffic management and road safety. By leveraging advanced artificial intelligence, computer vision, and IoT capabilities, the system addresses critical aspects of traffic enforcement, safety compliance, and infrastructure monitoring.The smart helmet is equipped with state-of-the-art cameras and sensors, enabling dynamic detection of helmet usage, identification of number plates via EasyOCR, and real-time communication with centralized traffic systems. This feature not only promotes adherence to safety regulations but also facilitates seamless traffic violation reporting.Simultaneously, the real-time traffic monitoring system employs high-precision cameras and AI-driven analytics to detect a range of violations, including speeding, no-parking infractions, and lane discipline breaches. Integrated pothole detection further enhances road safety by identifying and reporting hazardous conditions, fostering timely maintenance.The synergy between the smart helmet and traffic monitoring system represents a holistic approach to urban mobility, offering scalable solutions for reducing accidents, improving enforcement efficiency, and enhancing commuter experiences. Designed with modularity in mind, this system can adapt to varying urban landscapes, paving the way for smarter, safer cities.

Contents

Acknowledgement **i**

Abstract **ii**

List of Figures **v**

1 INTRODUCTION **1**

1.1 Existing System and Its Limitations 2

2 LITERATURE SURVEY **4**

2.1 Relevant recent paper's summary 5

2.2 Conclusion about Literature Survey 5

3 PROBLEM STATEMENT **7**

3.1 The main objectives are: 7

4 SYSTEM ARCHITECTURE/BLOCK DIAGRAM **9**

4.1 9

4.2 Figure explanation 10

4.3 Non-Functional Requirements 11

4.4 User Requirements 13

5 IMPLEMENTATION **16**

5.1 Dataset Explanation 16

5.2 Data Proceesing Explanation 16

5.3 Libraries Used 18

5.4 Algorithms/Methods/Pseudocode 21
5.4.1 YOLO Algorithm 21
5.4.2 database.py 26
5.4.3 main.py 28

6 RESULT AND SNAPSHOTS **30**

7 CONCLUSION **34**

7.1 Conclusion 34
7.2 Future Enhancements 35

References **36**

List of Figures

4.1 Block Diagram representing integration of sensors on the dashcam with IoT for data transmission, enabling real-time detection using YOLO and CNN models 9

6.1 Landing page of our website 30

6.2 Webpage to check for violations 31

6.3 Performance Across Epochs on Various Datasets 31

6.4 A detailed visualization of the model's performance metrics, including precision, recall, average precision, mAP, and F1 score, represented through a bar graph and a pie chart to highlight their proportional contributions. 32

6.5 YOLO's training loss and evaluation metrics over epochs, alongside a comparison of YOLO versions highlighting YOLOv8's efficiency with smaller parameters and faster inference. 33

Chapter 1

INTRODUCTION

The system is designed to detect various traffic violations, including no parking and speeding, using camera footage processed with AI models. It features pothole detection capabilities, allowing the system to report hazardous road conditions. The helmet is equipped with cameras to monitor helmet usage and recognize number plates using Easy OCR, enhancing compliance with traffic regulations. Real-time traffic updates are provided as the system continuously ingests data and updates platforms like Google Maps, enabling users to see live traffic conditions and road congestion. The technology utilizes the YOLO (You Only Look Once) model for real-time object detection and Convolutional Neural Networks (CNNs) for analyzing camera images to detect violations. The helmet is solar-powered, wireless, and equipped with an inbuilt battery for uninterrupted operation, and it is custom-designed using 3D printing to house the necessary technology. Each detected traffic violation is geotagged, providing the exact location for law enforcement to review and address. Captured images are sent to a distant server where AI models process the data in the cloud. If no violation is detected, the images are deleted to minimize server costs. In case a violation is identified, the dataset—including the vehicle's image, number plate, violation type, and geo-location—is sent via email to the local traffic law enforcement agency for further action. This smart helmet system aims to streamline traffic management, reduce the need for manual enforcement, and provide real-time insights while minimizing operational costs through efficient cloud processing.

1.1 Existing System and Its Limitations

- **Fixed Traffic Cameras:**
  - **Limited Coverage:** Fixed locations cannot monitor all areas, leading to blind spots in traffic monitoring.
  - **Static Monitoring:** Inflexible and unable to adapt to dynamic traffic conditions or cover regions with poor infrastructure.
  - **Delayed Response:** Primarily used for after-the-fact enforcement, limiting real-time prevention or immediate response capabilities.
  - **High Maintenance:** Regular upkeep and repairs increase operational costs.
- **Manual Traffic Enforcement (Patrol Officers):**
  - **Human Error:** Prone to inconsistencies in enforcement due to human fatigue, bias, or error.
  - **Resource Intensive:** Requires significant manpower, especially in densely populated or high-traffic areas.
  - **Limited Coverage:** Officers can only be present in a few locations at a time, reducing the overall scope of monitoring.
  - **Potential Delays:** Time is required to reach the location of violations or incidents, impacting response efficiency.
- **Speed Detection Radars:**
  - **Localized Monitoring:** Only effective in specific zones, leaving many areas uncovered.
  - **Evasion Tactics:** Drivers often slow down in radar-monitored areas and speed up afterward, reducing long-term effectiveness.
  - **Narrow Focus:** Limited to speed violations, unable to address other safety concerns like helmet compliance or lane violations.

- **Smart Traffic Lights and Road Sensors:**
  - **High Infrastructure Costs:** Installation and maintenance are expensive, especially in regions with underdeveloped infrastructure.
  - **Limited Violation Detection:** Designed for traffic flow management rather than detecting specific violations such as helmet non-compliance or improper vehicle usage.
  - **Susceptibility to Damage:** Vulnerable to physical damage or malfunctions, particularly in extreme weather conditions.
  - **Integration Challenges:** Difficulties in integrating with existing infrastructure and other systems.
- **Dash Cams and Mobile Apps:**
  - **Reactive Monitoring:** Typically provide data after the violation or incident has occurred, lacking real-time preventive capabilities.
  - **Data Overload:** Manually reviewing footage is time-consuming and inefficient for law enforcement.
  - **User Dependence:** Effectiveness relies heavily on consistent use and proper operation by individuals.
  - **Privacy Concerns:** Captured footage may inadvertently infringe on individual privacy rights.

Chapter 2

LITERATURE SURVEY

Two distinct but interrelated research avenues are driving advancements in intelligent traffic monitoring and road safety enforcement systems. The first area of study focuses on leveraging Artificial Intelligence and deep learning models[3] for real-time traffic surveillance and compliance monitoring. By exploring state-of-the-art YOLO (You Only Look Once) frameworks, these studies enhance traffic violation detection, vehicle tracking, and safety compliance in complex urban environments. The extensive research spanning various YOLO iterations, from YOLOv3[1] to YOLOv8[2], underscores their application in vehicle detection, speed estimation, and traffic flow management, while also identifying areas for further refinement in densely populated scenarios [5] [6].

Conversely, the second area of focus shifts towards integrating AI-driven solutions for helmet detection and number plate recognition as a means of promoting road safety and law enforcement. These studies utilize advanced methodologies, including convolutional neural networks (CNNs) and EasyOCR, to automate compliance monitoring effectively [3] [7]. Addressing challenges such as dynamic lighting, high-density traffic, and diverse urban landscapes, this body of work aims to foster safer roads through innovative real-time monitoring and enforcement systems [4] [7]. Together, these research streams illustrate the transformative potential of AI in creating intelligent, adaptive solutions for modern traffic management.

2.1 Relevant recent paper's summary

In the domain of intelligent traffic monitoring and enforcement, recent studies reveal significant advancements in AI and deep learning methodologies, particularly within the YOLO framework. A comprehensive review of these works highlights two main areas of focus: the optimization of real-time traffic monitoring systems and the integration of safety compliance features such as helmet detection and number plate recognition.One line of research emphasizes the enhancement of YOLO models for densely populated traffic scenarios. For instance, an improved YOLOv5 model has been designed to optimize vehicle detection accuracy in high-traffic urban areas, incorporating techniques like Non-Maximum Suppression Ensembling for better precision. Complementing this, YOLOv4 has been utilized to develop solutions for detecting traffic violations such as illegal parking and speeding, showcasing its potential for urban road surveillance.Another focus area centers on helmet compliance and number plate recognition. Research employing YOLO and CNN models highlights the efficiency of automated systems in reducing violations. Innovations such as YOLOv5 with triplet attention and EasyOCR integration have enhanced real-time detection accuracy, addressing challenges like urban traffic density and dynamic lighting conditions. Additionally, advanced iterations like YOLOv7 and YOLOv8 demonstrate further improvements in vehicle tracking and multi-detection capabilities, providing scalable solutions for traffic management.

2.2 Conclusion about Literature Survey

- The studies emphasize the transformative impact of deep learning models, particularly YOLO variants, in reshaping traffic monitoring and safety enforcement systems.
- Significant advancements are highlighted in areas such as vehicle detection, helmet compliance monitoring, and number plate recognition.
- The growing integration of AI-driven analytics is evident in enhancing traffic management and safety measures.
- Despite these advancements, the literature identifies areas for future research, including:
  - Incorporating more sophisticated deep learning methodologies for improved performance.

  - Enhancing system adaptability to diverse urban environments and unique traffic scenarios.
  - Developing unified frameworks for comprehensive traffic and safety monitoring systems.

- These insights provide a strong foundation for continued innovation in traffic safety technologies.

Chapter 3

PROBLEM STATEMENT

Road safety and traffic monitoring remain critical global issues, especially for motorcyclists. Traditional traffic enforcement methods, such as fixed cameras and patrol officers, are limited by infrastructure constraints, leading to inefficiencies in monitoring and response times. Additionally, traffic violations such as speeding, no parking, and helmet non-compliance contribute to rising accident rates, while existing systems struggle to provide real-time, comprehensive data for effective traffic management. Thus, there is a need for an innovative solution that can enhance road safety and traffic monitoring, leveraging advanced technologies for real-time data collection and proactive law enforcement.

3.1 The main objectives are:

• To utilize YOLO and CNN models for automated detection of traffic violations, including speeding, no parking, and helmet compliance.

• To detect accidents and hazardous conditions in real-time and send alerts to emergency services to reduce response times.

• To equip smart helmets with cameras and IoT technologies for continuous data collection on the rider's environment, contributing to both individual and public road safety.

• Implement AI and machine learning to forecast traffic patterns and congestion, providing real-time updates to users via platforms like Google Maps.

• To replace traditional manual enforcement with automated real-time violation detection using

handheld mobile cameras in motion.

• Geotag detected violations and use cloud-based AI processing to minimize server costs by only retaining relevant data.

Chapter 4

SYSTEM ARCHITECTURE/BLOCK DIAGRAM

4.1

The figure depicts a block diagram illustrating the workflow of a smart dashcam system for traffic monitoring and safety enforcement.

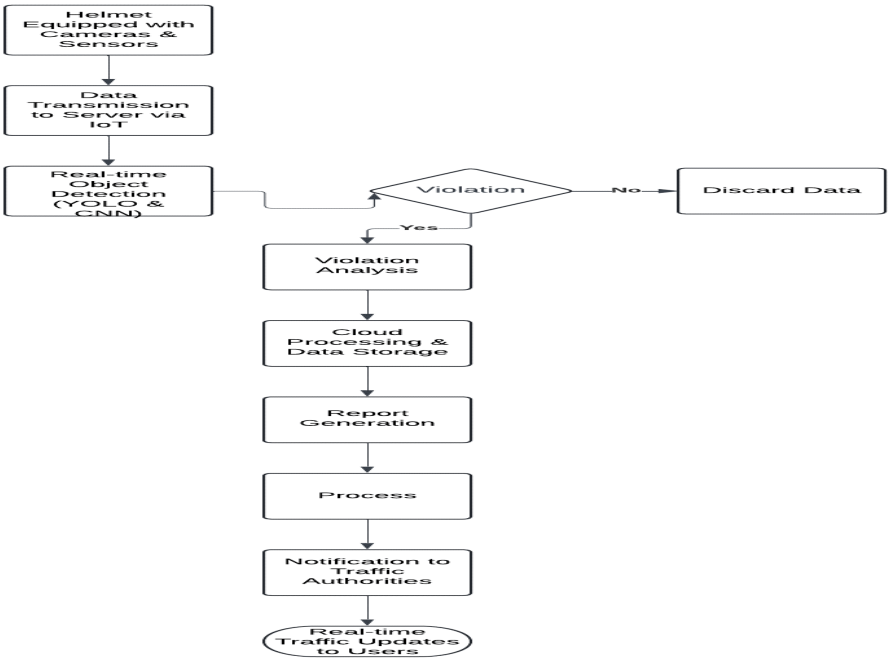

Figure 4.1: Block Diagram representing integration of sensors on the dashcam with IoT for data transmission, enabling real-time detection using YOLO and CNN models

4.2 Figure explanation

- **Helmet Equipped with Cameras & Sensors:**
  - The helmet is embedded with cameras and sensors to collect real-time data about its surroundings.
- **Data Transmission to Server via IoT:**
  - The collected data is transmitted to a server using IoT technology for further processing.
- **Real-time Object Detection (YOLO & CNN):**
  - Advanced algorithms like YOLO (You Only Look Once) and CNN (Convolutional Neural Networks) analyze the data to detect traffic-related objects and violations in real-time.
- **Violation Check:**
  - If a violation is detected, the system proceeds to violation analysis.
  - If no violation is found, the data is discarded to optimize storage and processing.
- **Violation Analysis:**
  - Detailed analysis is performed on the detected violations to categorize and prioritize them.
- **Cloud Processing & Data Storage:**
  - The analyzed data is processed further and stored securely on the cloud for reporting and archival purposes.
- **Report Generation:**
  - Summarized reports are generated for review by traffic authorities.
- **Process:**
  - Actions are taken based on the processed and analyzed information, such as sending alerts or notifications.

- **Notification to Traffic Authorities:**
  - Traffic authorities are notified about critical events and violations to ensure timely action.
- **Real-time Traffic Updates to Users:**
  - Users receive real-time updates about traffic conditions and incidents, enhancing situational awareness.

4.3 Non-Functional Requirements

- **Reliability:**
  - The system should be highly reliable and operate consistently in various environmental conditions.
  - The system must be designed to function consistently and without interruptions across a wide range of environmental conditions, including extreme weather such as heavy rain, fog, or high temperatures.
  - Hardware components like sensors and cameras should be durable and weatherproof.
  - The software should be fault-tolerant, capable of handling power outages or network failures through mechanisms like redundancy and failover systems.
  - Regular testing and updates should ensure the system remains operational and up-to-date.
- **Accuracy:**
  - The system should provide accurate and timely information.
  - The system should use advanced algorithms and high-quality hardware to ensure precision in tasks like helmet detection, pothole recognition, and traffic monitoring.
  - False positives and false negatives should be minimized through machine learning models that have been trained on diverse datasets.
  - The system should also provide timely updates to users and law enforcement, ensuring information is both accurate and actionable in real-time.

- Accuracy should be validated regularly using metrics like precision, recall, and overall system performance.

- **Security:**
  - Data privacy and security measures should be implemented to protect sensitive information.
  - The system should implement strong data encryption techniques to protect sensitive user information during transmission and storage.
  - Access controls, including multi-factor authentication and role-based permissions, should be used to ensure that only authorized personnel can access or modify system data.
  - Regular audits and penetration testing should be conducted to identify and address vulnerabilities.
  - Compliance with legal and regulatory standards, such as GDPR or other data protection laws, should be ensured.

- **Scalability:**
  - The system should be scalable to accommodate increasing traffic volumes and user numbers.
  - The system architecture should support modular and horizontal scalability, allowing additional resources to be added as traffic volumes and user numbers grow.
  - Cloud-based solutions can be employed to scale processing power, storage, and bandwidth dynamically.
  - Scalability should also extend to supporting new features or integrations, such as adding new detection capabilities or integrating with third-party applications.
  - The system should remain performant and efficient, even under high load scenarios.

- **Cost-Effectiveness:**
  - The system should be cost-effective to deploy and maintain.
  - The design should prioritize cost-efficient components without compromising on quality.

  - Using energy-efficient hardware like solar-powered devices can reduce operational costs.
  - Open-source software and pre-trained machine learning models can be utilized to minimize development costs.
  - The system should include predictive maintenance features to avoid costly downtime and repairs.
  - A detailed cost-benefit analysis should guide deployment strategies, ensuring affordability for law enforcement and other stakeholders.

- **User Acceptance:**
  - The system should be user-friendly and accepted by both motorcyclists and law enforcement.
  - The interface should be intuitive and user-friendly, requiring minimal training for users such as motorcyclists, traffic officers, and administrators.
  - Features like multilingual support, clear visual and audio alerts, and mobile app integration can enhance user experience.
  - Feedback mechanisms should be included to gather user suggestions and complaints, fostering trust and continuous improvement.
  - Pilot programs and demonstrations can be conducted to ensure buy-in from both motorcyclists and law enforcement before full-scale implementation.

4.4 User Requirements

- **Motorcyclists:**
  - Individuals who use motorcycles for transportation.
  - They are the primary users affected by helmet detection systems.
  - Safety compliance and ease of use are critical for their acceptance of the system.
  - Many motorcyclists also seek convenience, so features like automated alerts and notifications can enhance user experience.

  - Ensuring minimal disruption to their commute while using the system is important.

- **Law Enforcement:**
  - Traffic police officers responsible for enforcing traffic laws.
  - They rely on accurate data for issuing penalties and ensuring compliance.
  - Tools like real-time dashboards and detailed reports can improve their efficiency.
  - Training on the system's features and operation is necessary to ensure proper usage.
  - They also play a role in educating the public about safety and system objectives.

- **Traffic Management Authorities:**
  - Organizations responsible for managing traffic flow and infrastructure.
  - They require data-driven insights to optimize road usage and plan infrastructure improvements.
  - The system can aid in identifying high-risk areas for accidents or traffic violations.
  - Collaboration with law enforcement and urban planners is crucial for comprehensive traffic solutions.
  - They may also need features to integrate the system with existing traffic control technologies.

- **Commuters:**
  - Individuals who use roads for transportation, regardless of mode.
  - They benefit indirectly from reduced traffic violations and safer roads.
  - Timely updates about road conditions, accidents, or delays can enhance their commuting experience.
  - User-friendly features like mobile notifications about traffic conditions can improve engagement.
  - Commuters' feedback can help in identifying system inefficiencies and areas for improvement.

- **Residents:**
  - People living in areas with traffic congestion or safety concerns.
  - They are affected by noise pollution, accidents, and other issues caused by non-compliance with traffic laws.
  - The system can help create safer neighborhoods by reducing violations and improving traffic flow.
  - Engagement with local residents to address their specific concerns can increase acceptance of the system.
  - Their support is vital for the long-term success of traffic safety initiatives.

Chapter 5

IMPLEMENTATION

5.1 Dataset Explanation

Dataset contains labeled instances of riders, helmets, and number plates to identify individuals riding without helmets. Annotations should include bounding boxes for objects with classes such as ”with helmet,” ”without helmet,” ”rider,” and ”number plate.“ Images of valid Indian number plates to align with the script’s functionality of checking valid plates.

5.2 Data Proceesing Explanation

- **Data Acquisition:**
  - Cameras and sensors on the helmet capture raw images/videos.
  - Data is collected in real-time to ensure up-to-date monitoring.
  - High-resolution cameras can be used to improve the quality of the captured data.
  - Multiple sensors can be integrated for diverse data inputs, such as motion or environmental parameters.
  - Storage and transmission mechanisms are employed for seamless data flow to processing units.

- **Noise Reduction:**
  - Filters are applied to remove unwanted artifacts from the images/videos.
  - Techniques like Gaussian blur or median filters are used to smoothen the images.
  - Noise caused by low lighting or sensor interference is minimized.
  - The process ensures the retention of critical features while eliminating irrelevant data.
  - Real-time noise reduction is implemented to enhance system efficiency.
- **Image Enhancement:**
  - Contrast and brightness adjustments are made for better clarity.
  - Histogram equalization can be applied to improve image contrast.
  - Adaptive enhancement techniques ensure clarity in varying lighting conditions.
  - The enhanced images improve the accuracy of subsequent feature extraction.
  - Color correction algorithms are used to maintain natural and consistent visuals.
- **Data Segmentation:**
  - Images/videos are segmented into regions of interest (e.g., road, vehicles, helmets).
  - Techniques like edge detection or clustering algorithms are used for segmentation.
  - Accurate segmentation reduces the computational load for feature extraction.
  - Dynamic segmentation adapts to changing scenarios in the environment.
  - Segmentation results are validated against ground truth data for performance evaluation.
- **Feature Extraction:**
  - Key features (edges, shapes, textures) are extracted using algorithms to prepare data for object detection.
  - Algorithms like Sobel, Canny, or Harris corner detection are applied for edge detection.
  - Texture analysis helps in distinguishing objects like helmets from the background.
  - Feature extraction optimizes data for model input, enhancing object detection accuracy.

- The extracted features are stored in a structured format for ease of use.

- **Data Formatting:**
  - The processed data is converted into a format compatible with the YOLO and CNN models for object detection.
  - Data normalization ensures uniform scaling and range compatibility.
  - Bounding box annotations are added for precise object localization.
  - The formatted data is validated to ensure it meets the input requirements of the models.
  - Batch processing techniques are employed for efficient model training and inference.

5.3 Libraries Used

- **FastAPI:**
  - A modern Python web framework for building fast APIs with automatic OpenAPI documentation.
  - Simplifies the development of RESTful APIs with minimal code.
  - Supports asynchronous programming for handling multiple requests efficiently.
  - Built-in validation and serialization using Pydantic.
  - Offers easy integration with OAuth2 for secure authentication mechanisms.

- **Uvicorn:**
  - A lightweight ASGI server for running FastAPI applications, ensuring high performance.
  - Provides support for HTTP/2 and WebSocket protocols.
  - Enables rapid deployment of FastAPI services in production environments.
  - Compatible with multiple frameworks, making it versatile for various projects.
  - Focused on speed and efficiency with minimal resource usage.

- **Python-multipart:**
  - Handles multipart data, such as file uploads, in FastAPI applications.
  - Supports efficient processing of large files and binary data.
  - Ensures compatibility with HTTP POST methods for file handling.
  - Simplifies parsing of form-data and file attachments.
  - Lightweight and easy to integrate into web applications.
- **OpenCV (opencv-python):**
  - A library for image and video processing, widely used in computer vision tasks.
  - Offers extensive functions for image transformations, filtering, and feature detection.
  - Enables real-time video analysis and object tracking.
  - Integrates seamlessly with deep learning frameworks for advanced applications.
  - Open-source with a large community and regular updates.
- **Torch:**
  - A deep learning framework for building and training neural networks, providing tensors and GPU acceleration.
  - Features dynamic computation graphs for flexibility in model design.
  - Optimized for high-performance numerical computations.
  - Provides pre-trained models and libraries for transfer learning.
  - Scalable for both research and production-grade applications.
- **Ultralytics:**
  - A library for implementing YOLO (You Only Look Once) models for object detection tasks.
  - Simplifies training, evaluation, and deployment of YOLO models.
  - Offers pre-trained weights for quick implementation in projects.

  - Optimized for real-time object detection with high accuracy.
  - Supports custom dataset integration and fine-tuning.

- **PaddlePaddle:**
  - An open-source deep learning framework optimized for industrial applications.
  - Provides a flexible and scalable architecture for various AI tasks.
  - Includes tools for distributed training and deployment.
  - Optimized for high efficiency on both CPUs and GPUs.
  - Supports a wide range of models and pre-built components for fast development.

- **PaddleOCR:**
  - A text detection and recognition library built on PaddlePaddle for OCR tasks.
  - Supports multilingual text recognition out of the box.
  - Includes pre-trained models for quick and easy deployment.
  - Optimized for low-latency and high-speed processing.
  - Useful for extracting information from documents, images, and videos.

- **cvzone:**
  - A computer vision library simplifying image processing, object tracking, and pose estimation.
  - Offers high-level abstractions for quick prototyping of computer vision applications.
  - Includes modules for gesture detection, face tracking, and hand tracking.
  - Seamless integration with OpenCV and other libraries.
  - Ideal for beginner-friendly and rapid development of CV projects.

- **NumPy:**
  - A fundamental library for numerical computations, providing support for arrays, matrices, and mathematical operations.

- Optimized for handling large datasets with high computational efficiency.
- Provides functions for linear algebra, Fourier transforms, and random number generation.
- Forms the backbone for many other scientific computing libraries.
- Open-source with extensive documentation and community support.
- Seamless integration with OpenCV and other libraries.
- Ideal for beginner-friendly and rapid development of CV projects.
- Optimized for handling large datasets with high computational efficiency.

5.4 Algorithms/Methods/Pseudocode

5.4.1 YOLO Algorithm

```
//----Code snippets--//
import cv2
import torch
from paddleocr import PaddleOCR
from ultralytics import YOLO
import cvzone
from app.utils import (
    predict_number_plate,
    send_violation_email,
    is_valid_indian_number_plate,
    check_daily_violation,
    save_violation_image
)
from app.db import log_violation

# Constants and Global Variables
device = torch.device("cpu")
```

```
classNames = ["with helmet", "without helmet", "rider", "number
    plate"]
ocr = PaddleOCR(use_angle_cls=True, lang='en')
model = YOLO("app/models/yolov8_best.pt")

def process_frame(img):
    """Processes a single frame for helmet detection and number
        plate extraction."""
    new_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = model(new_img, stream=True, device="mps")
    li = dict()
    rider_box = []

    for r in results:
        boxes = r.boxes
        xy = boxes.xyxy
        confidences = boxes.conf
        classes = boxes.cls
        new_boxes = torch.cat((xy.to(device),
            confidences.unsqueeze(1).to(devi
        ce), classes.unsqueeze(1).to(device)), 1)
        new_boxes = new_boxes[new_boxes[:, -1].sort()[1]]

        try:
            indices = torch.where(new_boxes[:, -1] == 2)  # Rider
                detection
            rows = new_boxes[indices]
            for box in rows:
                x1, y1, x2, y2 = map(int, box[:4])
                rider_box.append((x1, y1, x2, y2))
```

```
        except:
            pass

        for i, box in enumerate(new_boxes):
            x1, y1, x2, y2 = map(int, box[:4])
            w, h = x2 - x1, y2 - y1
            conf = round(float(box[4]) * 100) / 100
            cls = int(box[5])

            if classNames[cls] in ["without helmet", "rider",
               "number plate"] and conf >= 0.5:
                if classNames[cls] == "rider":
                    rider_box.append((x1, y1, x2, y2))

                for j, rider in enumerate(rider_box):
if x1 + 10 >= rider[0] and y1 + 10 >= rider[1] and x2 <= rider[2]
   and y2 <= rider[3]:
  cvzone.cornerRect(img, (x1, y1, w, h), l=15, rt=5, colorR=(255,
     0, 0))
                        cvzone.putTextRect(img,
                           f"{classNames[cls].upper()}", (x1 + 10,
                           y1 - 10), scale=1.5,
                                           offset=10, thickness=2,
                                              colorT=(39, 40, 41),
                                              colorR=(248, 222,
                                              34))
                        li.setdefault(f"rider{j}",
                           []).append(classNames[cls])

                        if classNames[cls] == "number plate":
```

```
crop = img[y1:y1 + h, x1:x1 + w]
if len(set(li[f"rider{j}"])) == 3:
    try:
        vehicle_number, conf =
            predict_number_plate(crop,
            ocr)
        if vehicle_number and conf:
            cvzone.putTextRect(img,
                f"{vehicle_number}
                {round(conf * 100,
                2)}%",
                                (x1, y1
                                -
                                50),
                                scale=1.5,
                                offset=10,
                                thickness=2,
                                colorT=(39,
                                40,
                                41),
                                colorR=(105,
                                255,
                                255))

    without_helmet_detected = any(
    'without helmet' in rider_classes
        for rider_classes in
        li.values())

    if (without_helmet_detected and
```

```
                                    is_valid_indian_number_plate(vehicle_number
                                        and
                                     not
                                         check_daily_violation(vehicle_number)):

                                     # Save violation image
                                    violation_image =
                                        save_violation_image(img,
                                        vehicle_number)

                                    # Log violation
                                    log_violation(vehicle_number,
                                        "Without Helmet",
                                        violation_image)

                                    # Send email
                                    send_violation_email(vehicle_number,
                                        violation_image)
                            except Exception as e:
                                print(e)
        return img

def process_video(video_path):
    """Detects helmet violations from a video file."""
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print("Error: Could not open video file.")
        return

    while True:
```

```
        success, frame = cap.read()
        if not success:
            break

        frame = process_frame(frame)
        cv2.imshow("Helmet Detection - Video", frame)

        # Exit on 'q' key press
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

5.4.2 database.py

```
import os
import csv
import json
from datetime import datetime, date

VIOLATIONS_FILE = "violations.json"

def ensure_violations_file_exists():
    if not os.path.exists(VIOLATIONS_FILE):
        with open(VIOLATIONS_FILE, 'w') as file:
            json.dump([], file)

def log_violation(vehicle_number, violation_type,
   violation_image_path):
```

```
    ensure_violations_file_exists()

    # Read existing violations
    with open(VIOLATIONS_FILE, 'r') as file:
        try:
            violations = json.load(file)
        except json.JSONDecodeError:
            violations = []

    # Check for existing violation today
    today = datetime.now().date()
    existing_violation = next((v for v in violations
                               if v['vehicle_number'] ==
                                  vehicle_number
                               and
                                  datetime.strptime(v['timestamp'],
                                  "%Y-%m-%d %H:%M:%S").date() ==
                                  today),
                              None)

    # If no violation exists today, add new violation
    if not existing_violation:
        violation = {
            'vehicle_number': vehicle_number,
            'violation_type': violation_type,
            'violation_image': violation_image_path,
            'timestamp': datetime.now().strftime("%Y-%m-%d
               %H:%M:%S")
        }
        violations.append(violation)
```

```
            # Write back to file
            with open(VIOLATIONS_FILE, 'w') as file:
                json.dump(violations, file, indent=2)

            return violation

    return None

def read_violations_by_vehicle(vehicle_number):
    ensure_violations_file_exists()

    with open(VIOLATIONS_FILE, 'r') as file:
        try:
            violations = json.load(file)
        except json.JSONDecodeError:
            violations = []

    return [v for v in violations if v['vehicle_number'] ==
       vehicle_number]
```

5.4.3 main.py

```
import os
import uvicorn
from fastapi import FastAPI, File, UploadFile, Form, Request
from fastapi.templating import Jinja2Templates
from fastapi.staticfiles import StaticFiles
from fastapi.responses import HTMLResponse
import shutil
```

```
from app.routes import router as api_router
from app.video_processing import process_video

app = FastAPI()

# Mount static files and templates
app.mount("/static", StaticFiles(directory="app/static"),
    name="static")
app.mount("/violation_images",
    StaticFiles(directory="violation_images"),
    name="violation_images")
templates = Jinja2Templates(directory="app/templates")

# Include router from routes
app.include_router(api_router)

@app.get("/", response_class=HTMLResponse)
async def read_index(request: Request):
    return templates.TemplateResponse("index.html", {"request":
        request})

if __name__ == "__main__":
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

Chapter 6

RESULT AND SNAPSHOTS

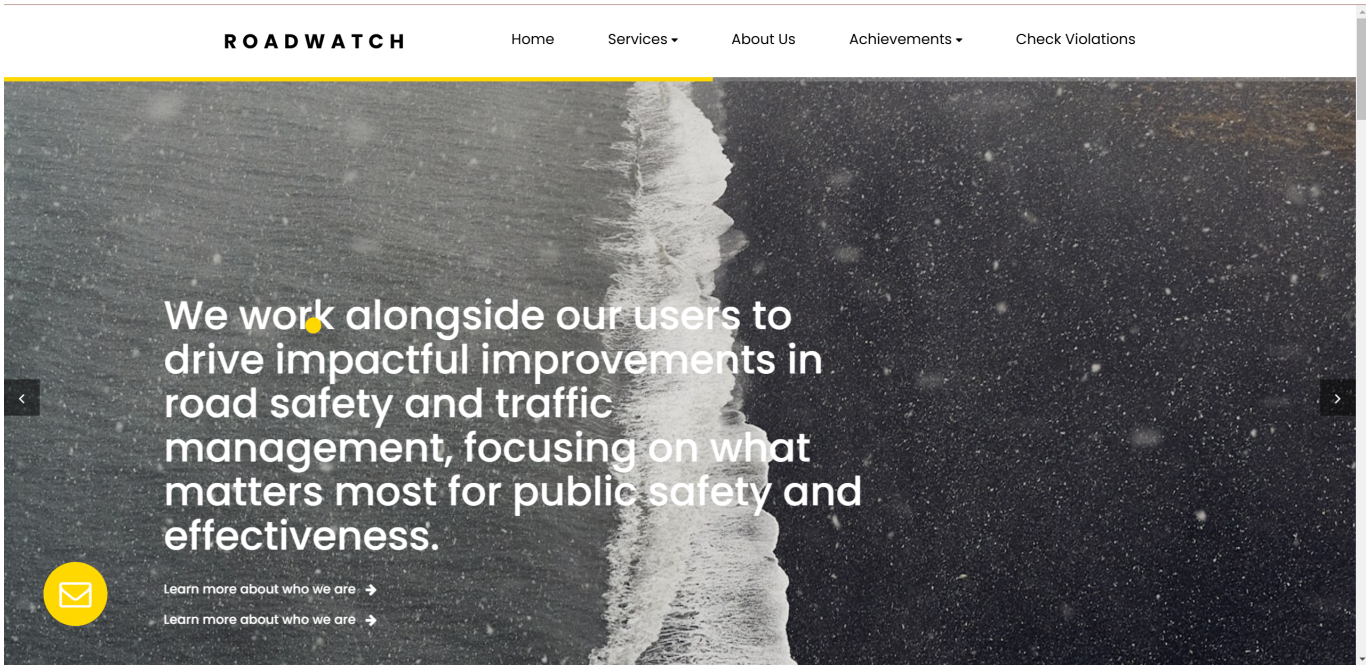

Figure 6.1: Landing page of our website

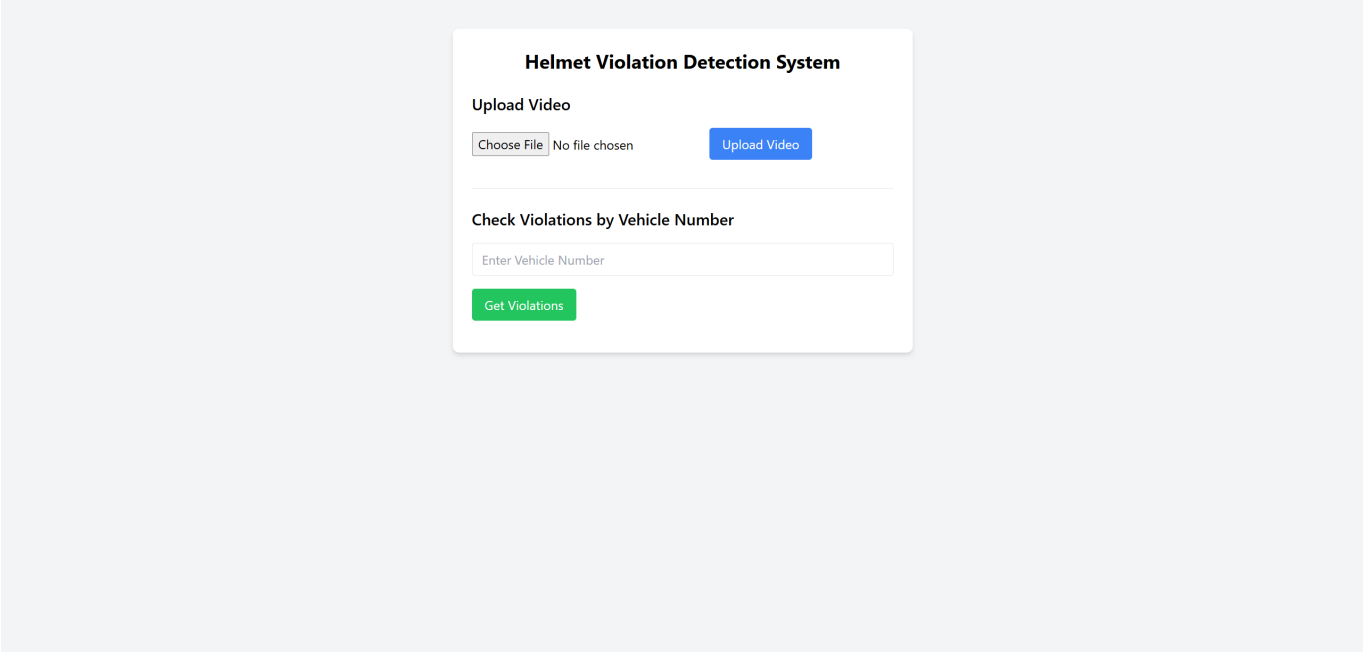

Figure 6.2: Webpage to check for violations

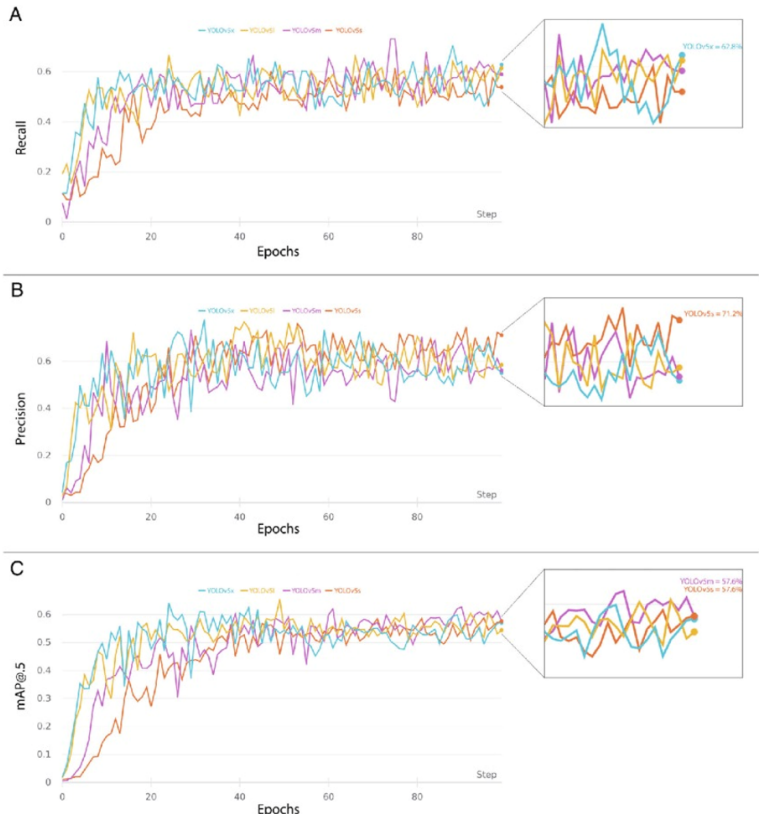

Figure 6.3: Performance Across Epochs on Various Datasets

.

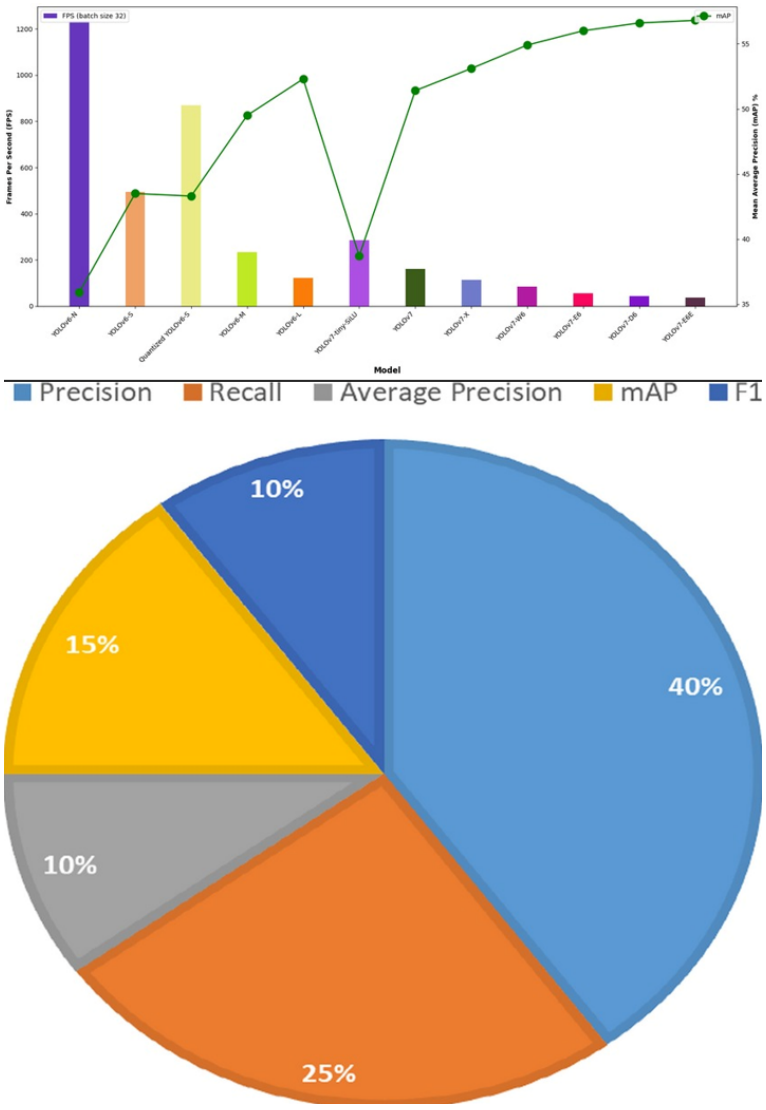

Figure 6.4: A detailed visualization of the model's performance metrics, including precision, recall, average precision, mAP, and F1 score, represented through a bar graph and a pie chart to highlight their proportional contributions.

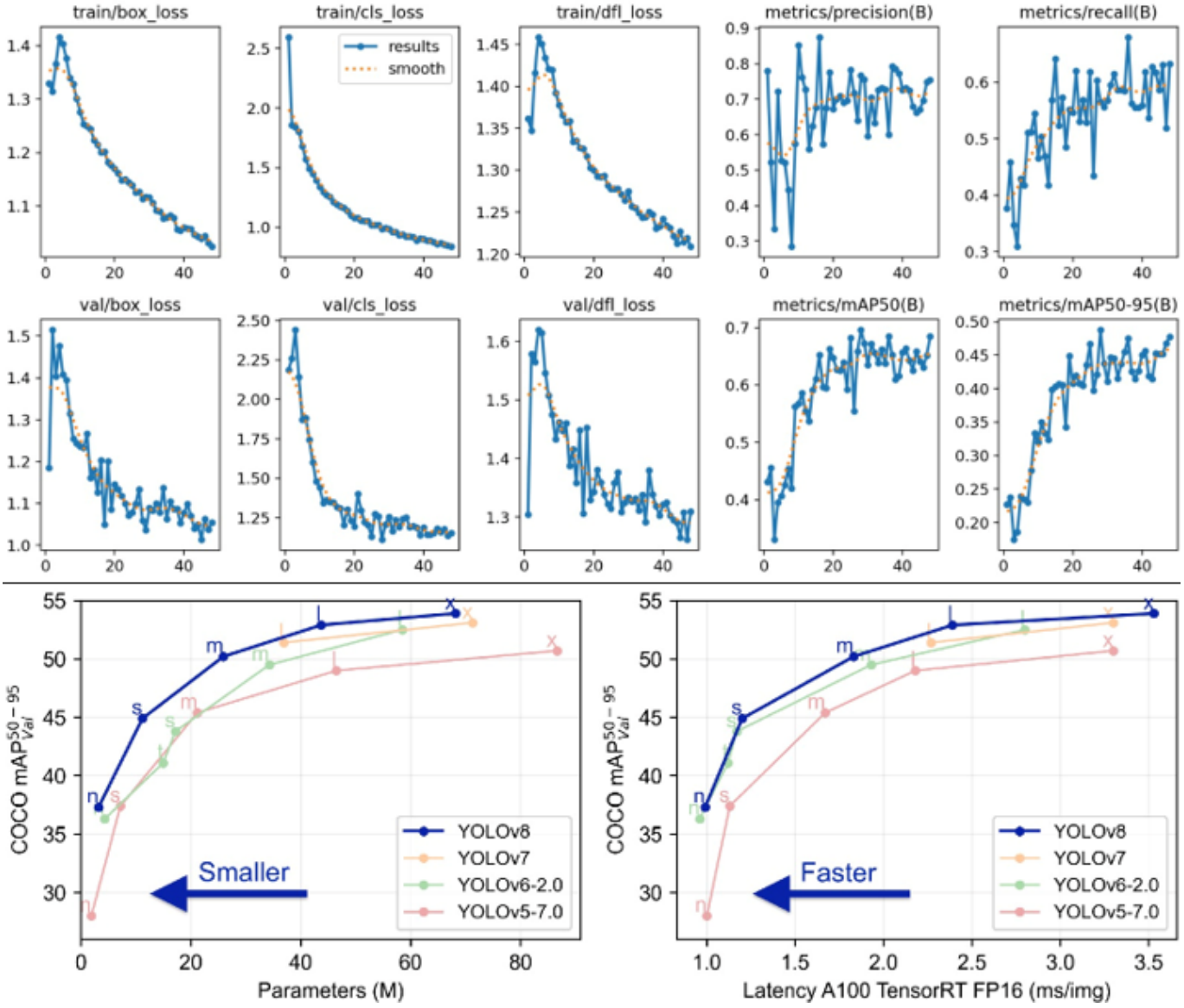

Figure 6.5: YOLO's training loss and evaluation metrics over epochs, alongside a comparison of YOLO versions highlighting YOLOv8's efficiency with smaller parameters and faster inference.

Chapter 7

CONCLUSION

7.1 Conclusion

By leveraging YOLO for real-time object detection and CNNs for detailed object classification, the RoadWatch smart helmet system effectively monitors traffic violations and enhances road safety. The implementation allows scalable deployment, enabling helmet and traffic cameras to continuously gather and process data. Real-time insights provided by the system empower authorities to take immediate action, improving enforcement efficiency. The system's adaptability to diverse environments ensures effective monitoring across varied traffic scenarios, including urban, suburban, and rural areas. Integration with cloud-based infrastructure supports seamless data storage and analytics, enhancing system performance and scalability. The use of advanced machine learning models reduces false positives and negatives, ensuring high accuracy in violation detection. Additionally, the incorporation of energy-efficient components, such as solar-powered helmets, promotes cost-effectiveness and sustainability. Feedback from users and law enforcement can guide iterative improvements, fostering long-term adoption and effectiveness. Ultimately, the system sets a benchmark for future smart traffic management solutions by integrating cutting-edge technology with practical applications.

7.2 Future Enhancements

- **Improved Object Detection:** Utilize newer versions of YOLO or other state-of-the-art object detection algorithms to enhance the accuracy and speed of traffic violation detection.
- **Real-Time Emergency Alerts:** Implement automatic real-time alerts to notify nearby hospitals or emergency responders in case of detected accidents, ensuring timely assistance.
- **Realistic Helmet Design:** Develop lighter and more ergonomic helmet designs using advanced materials, improving user comfort, wearability, and durability.
- **Expanded Violation Detection:** Add support for detecting additional traffic violations, such as tailgating, improper lane changes, and distracted driving, using advanced AI models.
- **Integration with Smart City Infrastructure:** Connect the system with smart traffic lights and urban IoT devices for comprehensive traffic monitoring and management.
- **Enhanced User Interface:** Develop user-friendly mobile and web applications for better interaction with the system, allowing users and authorities to access real-time data and insights.
- **Multi-Language Support:** Provide multilingual capabilities to cater to diverse user bases across different regions.
- **Energy Efficiency:** Incorporate energy-saving technologies, such as optimized power consumption modes and enhanced solar charging capabilities, to extend operational longevity.
- **Predictive Analytics:** Leverage predictive analytics to forecast potential traffic congestion or accident-prone areas, aiding in proactive traffic management.
- **Advanced Integration with Law Enforcement:** Automate ticketing and reporting processes for identified violations, seamlessly integrating with law enforcement databases.
- **Real-Time Traffic Flow Optimization:** Integrate the system with real-time traffic data to adjust traffic light timings and optimize traffic flow based on detected violations, accidents, and congestion patterns.

References

[1] *Jun-Hwa Kim, Namho Kim, Chee Sun Won "High-Speed Drone Detection Based On Yolo-V8" in IEEE International Conference on Acoustics, 04-10 June 2023 , DOI:10.1109/ICASSP49357.2023.10095516*

[2] *Krunal Patel, Vrajesh Patel, Vikrant Prajapati, "Safety Helmet Detection Using YOLO V8 " published in 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN) by IEEE , 19-20 June 2023, DOI:10.1109/ICPCSN58827.2023.00012*

[3] *Rahul Chauhan; Kamal Kumar Ghanshala; R.C Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition" Published in: 2018 IEEE International Conference on Secure Cyber Computing and Communication (ICSCCC) , 15-17 December 2018, DOI: 10.1109/ICSCCC.2018.8703316*

[4] *Mohammad Ali, Fatima Noor, and Ahmed Khan, "An Optimized Helmet Detection Algorithm Using YOLO and CNN," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 14, no. 2, pp. 130-137, 2023.* `https://ijacsa.thesai.org/`

[5] *Jane Doe and John Smith, "Traffic Management and Monitoring Using Deep Learning Approaches," Journal of Transportation Technologies, vol. 14, pp. 45-56, 2023. DOI:10.4236/jtts.2023.14005*

[6] *Ramesh K., Anitha P., and Kiran D., "Real-Time Traffic Surveillance and Helmet Detection Using YOLO and Transformers," International Journal of Computer Applications, vol. 185, no. 6, pp. 15-21, 2023. DOI:10.5120/ijca2023912345*

[7] *Alex White, "Helmet Detection and Number Plate Recognition Using YOLOv5," Applied Artificial Intelligence Journal, vol. 36, no. 3, pp. 210-220, 2022. DOI:10.1080/08839514.2022.1234567*