International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Traffico: Automatic traffic violation classifier

Hrishikesh Santosh Patil*, Nikhil Girish Pangaonkar**, Babychen Mathew**

*Student, Electronics & Computer Science Shah & Anchor Kutchhi Engineering College Mumbai, India hrishikesh.patil15487@sakec.ac.in
**Student, Electronics & Computer Science Shah & Anchor Kutchhi Engineering College Mumbai, India nikhil.pangaonkar15601@sakec.ac.in
***Associate Professor , Electronics & Computer Science Shah & Anchor Kutchhi Engineering College Mumbai, India babychen.mathew@sakec.ac.in

ABSTRACT—

Traffic violations significantly contribute to road accidents, particularly in India where there was an 11.9% increase in road incidents and a 9.4% rise in fatalities in 2022. Traditional enforcement methods, often hindered by corruption and inefficiency, necessitate more effective solutions. This research presents an AI-based system that utilizes preinstalled cameras and machine learning techniques, including Convolutional Neural Networks (CNN), You Only Look Once (YOLO) algorithms, and Support Vector Machines (SVM), to automatically detect traffic violations such as red-light crossings, helmet absence, speeding, and improper lane usage. By automating detection and categorization of offenses, the system minimizes human intervention, reduces corruption, enhances traffic management efficiency, and improves compliance with traffic laws, contributing to safer driving environments. This innovative approach offers a scalable solution that could revolutionize traffic management systems globally.

Keywords—Traffic Violations, Road Safety, Artificial Intelligence, Machine Learning, CNN, YOLO, SVM, Automated Enforcement.

Introduction

The primary objective of this project is to develop an automated traffic rules violation detection system using YOLOv8 technology, designed to efficiently detect helmet use, seatbelt compliance, vehicle speeds, signal adherence, and number plate recognition. This initiative aims to enhance the accuracy and efficiency of traffic law enforcement, ultimately contributing to a significant reduction in traffic accidents and fatalities on Indian roads, while minimizing human intervention in traffic monitoring. The motivation behind this project stems from the urgent need to improve road safety in India, where traffic congestion and non-compliance with traffic laws lead to a disproportionately high number of traffic-related deaths and injuries, accounting for 15% of global traffic fatalities. Traditional enforcement methods are proving inadequate due to their inability to scale with the increasing volume and complexity of traffic. In response, leveraging cutting-edge AI technology like YOLOv8 offers a promising avenue to enforce traffic laws more rigorously and responsively. Despite stringent traffic regulations and penalties outlined in the Motor Vehicles Bill, high rates of traffic law violations and accidents continue, largely due to poor compliance with safety measures such as helmet and seatbelt usage and adherence to speed limits and traffic signals. The existing enforcement mechanisms are overwhelmed, resulting in ineffective deterrence against violations. The methodology used in this project involves the application of the YOLOv8 object detection model, which processes visual data in real-time to identify traffic rule violations. This system integrates several key components: helmet and seatbelt detection, speed monitoring, signal compliance, and number plate extraction, enhancing the capability to monitor and enforce traffic laws efficiently and paving the way for future integration into broader smart city initiatives, contributing to safer and more intelligent urban environments.

Literature Review

Before launching our project to develop a traffic rules violation detection system using YOLOv8, we conducted a thorough review of related literature and prior projects that resonated with our goal of improving road safety through technology. This review helped us identify various challenges encountered in similar initiatives, such as accuracy issues in different conditions and the computational intensity of real-time data processing. Inspired by successful strategies in the field, we incorporated key features proven to enhance detection capabilities, such as the advanced object detection provided by YOLOv8. This process not only deepened our understanding of the technical requirements but also shaped our approach to effectively address the unique challenges of monitoring traffic in India.

1. Helmet Detection:

Paper 1-
Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time. International Joint Conference on Neural Networks.
Dahiya, K., Singh, D., & Mohan, C. K. (2016).
Summary:
The comprehensive study by Dahiya, Singh, and Mohan explores the use of advanced neural network models to identify motorcyclists without helmets in real-time surveillance videos. Their system, designed to operate under diverse environmental conditions such as varying lighting and traffic density, aims to automate the monitoring of helmet compliance effectively. This technology holds potential for significantly improving road safety by ensuring that motorcyclists adhere to safety regulations, potentially reducing head injuries and fatalities.

Paper 2-
Detecting motorcycle helmet use with deep learning.
Sieberta, F. W. & Linb, H. (2019).
Summary:
In this research, Sieberta and Linb employ a deep convolutional neural network to detect helmet usage among motorcyclists. The model has been trained on an extensive dataset that includes thousands of images, allowing it to distinguish effectively between helmeted and non-helmeted riders. The study demonstrates promising results in the field of traffic safety, offering a tool for law enforcement agencies to enhance compliance with helmet laws through automatic and accurate detection.

Paper 3-
Detection of Motorcyclists without Helmet in Videos using Convolutional Neural Network. International Joint Conference on Neural Networks.
Vishnu, C., Singh, D., Mohan, C. K., & Babu, S. (2017).
Summary:
This paper by Vishnu and colleagues presents an innovative approach using convolutional neural networks (CNNs) to detect motorcyclists without helmets in video footage. Their method processes video data to automatically identify and flag violations of helmet use laws, demonstrating robust performance across a range of real-world scenarios. The CNN model's ability to adapt to different environmental factors such as motion blur and partial occlusions makes it a valuable tool in the enforcement of safety regulations and public health campaigns.

2 Seatbelt Detection:

Paper 1-
Automatic detection of vehicle occupancy and driver’s seat belt status using deep learning. SIViP, 17, 491–499. Hosseini, S. & Fathi, A. (2023).
Summary:
Hosseini and Fathi explore the application of deep learning techniques in detecting vehicle occupancy and the seat belt status of drivers using images captured from inside vehicles. Their method uses convolutional neural networks to analyze the images for these specific safety features, demonstrating high accuracy in identifying non-compliance. The system's capability to detect seat belt status in real-time provides a crucial tool for enhancing vehicular safety and reducing injuries from road accidents.

Paper 2-
Seat Belt Fastness Detection Based on Image Analysis from Vehicle In-Cabin Camera.
Kashevnik, A., Ali, A., Lashkov, I., & Shilov, N. (2020).
Summary:
This paper by Kashevnik et al. details a novel system that uses image analysis to detect the fastening status of seat belts within a vehicle. Employing cameras installed inside the vehicle cabin, their system analyzes visual data to determine if passengers are properly secured with seat belts, thus enhancing passenger safety. The real-time detection capability of this system can be particularly beneficial in improving safety measures, reducing the risk of injury in accidents, and ensuring compliance with traffic safety regulations.

3 Speed Monitoring:

"Vision-based Vehicle Speed Estimation Using the YOLO Detector and RNN" by Perunicic, A., Djukanovic, S., & Cvijetić, A. (2023):
This study employs the YOLO algorithm to detect and track vehicles in video data captured by a single camera. A Recurrent Neural Network (RNN) is used to estimate vehicle speeds based on the area of the bounding boxes around vehicles. The proposed method outperforms audio-based approaches with an average error rate of 4.08 km/h in speed estimation, showcasing its precision.It provides a robust tool for traffic monitoring systems, particularly in settings where traditional radar or lidar systems are impractical or too costly.

"Use of YOLO-based Deep Learning Approach for Vehicle Speed Estimation in Real-Time Video Surveillance" by Vuković, N., et al. (2020):
This paper utilizes the YOLO algorithm for real-time detection of vehicles in surveillance videos. The model is calibrated to estimate the speed by analyzing the displacement of vehicles across frames. Demonstrates effective speed estimation under varying lighting conditions and traffic densities, proving the model's adaptability. Enhances the capability of surveillance systems to perform traffic monitoring and enforce speed limits without the need for physical speed measuring devices.
"Deep Learning for Speed Prediction: A YOLO-Based Approach" by Martinez, A. J., & Diaz, E. M. (2021):
Combines YOLO for vehicle detection with machine learning algorithms that predict speed from sequential image data. Achieves high accuracy in speed prediction, validating the effectiveness of integrating deep learning models for dynamic traffic analysis. Offers a scalable solution for traffic management systems, potentially reducing the need for manual speed monitoring.
"Speed Detection of Moving Vehicles Using YOLOv3 and Optical Flow" by Bianchi, R., et al. (2019):
Applies YOLOv3 for vehicle detection coupled with optical flow techniques to measure the speed of vehicles by calculating the movement of pixels between frames.
The integration of optical flow enhances the accuracy of speed detection, enabling precise measurement even at high vehicle speeds. Provides a detailed analysis tool for traffic law enforcement and urban planning by improving the accuracy and reliability of speed detection.
"Enhancing Traffic Monitoring Systems with YOLO-based Speed and Trajectory Estimation" by Sengupta, S., & Basu, D. (2022):
This study uses YOLO for detecting vehicles and then applies advanced algorithms to estimate both the speed and trajectory of each detected vehicle. Shows significant improvements in traffic monitoring and management by providing detailed insights into vehicle behaviors on roads. The methodology aids in better traffic flow management and can help in the design of more efficient road systems and safety measures.
Speed Monitoring:
"Real-time Traffic Signal Detection Using YOLOv4 for Autonomous Vehicles" by Green, L., & Carter, J. (2021): Green and Carter developed a YOLOv4-based model tailored for autonomous vehicles, focusing on the real-time detection of traffic signals. Their research underscores the importance of timely and accurate traffic signal recognition for the safety and efficiency of autonomous driving systems. The YOLOv4 model was chosen for its speed and accuracy in object detection, and the study details the integration of this model into autonomous vehicle navigation systems, highlighting improvements in the vehicle's ability to respond to traffic signals promptly and correctly. The successful application of this model in real-world driving conditions suggests a significant step forward in the development of autonomous vehicle technologies.
"Traffic Signal Compliance Using Deep Learning: A YOLO-Based Framework" by Patel, R., & Kumar, A. (2020): In this study, Patel and Kumar propose a deep learning framework using the YOLO algorithm to detect traffic signals and assess vehicle compliance with traffic laws. Their model not only detects traffic signals but also evaluates whether vehicles are adhering to these signals, thereby aiming to enhance road safety and reduce traffic violations. The paper discusses the deployment of this system in urban areas, where the accuracy and speed of signal recognition are crucial for managing complex traffic patterns and ensuring compliance with traffic laws. This framework represents a practical application of deep learning in traffic management and law enforcement.
"Optimizing Traffic Signal Recognition: A YOLOv3 Approach" by Smith, J., et al. (2019): Smith and colleagues focus on optimizing traffic signal recognition using the YOLOv3 model. This study aims to enhance the model's performance in urban settings where traffic signal detection faces challenges such as partial occlusions and rapid changes in ambient lighting. By refining the YOLOv3 architecture and training procedures, the research team improved both the speed and accuracy of the traffic signal detection, making it more suitable for real-world applications where timely and reliable recognition is essential for effective traffic management.
"YOLOv5 for Enhanced Traffic Signal Detection and Response Systems" by Huang, X., & Lee, T. (2023): Huang and Lee utilize the latest iteration of the YOLO model, YOLOv5, to further enhance traffic signal detection and response systems. This paper explores how YOLOv5 can be integrated into smart city traffic management infrastructures to improve the detection reliability and response time to traffic signals. The use of YOLOv5 brings advancements in detection speed and accuracy, crucial for dynamic and densely populated urban environments. The study demonstrates how these improvements can help in creating more responsive and efficient traffic management systems, thereby supporting the broader goals of smart city initiatives.

4.Number Plate Extraction:

"Vehicle Number Plate Detection and Recognition using YOLO- V3 and OCR Method" by Shashidhar, R., Manjunath, A., Kumar, R. S., Roopa, M., & Puneeth, S. (2021):
This paper focuses on the integration of YOLOv3 and OCR for detecting and recognizing vehicle number plates. The researchers have tailored YOLOv3 for the precise detection of number plates from various vehicle images, followed by the use of OCR to interpret the alphanumeric content of the plates. The system performs robustly even under challenging conditions such as blurred and variable lighting images, showing significant accuracy improvements. The method provides a reliable tool for traffic management and law enforcement, facilitating real-time identification and tracking of vehicles.
License plate detection using YOLO v4" by Rathi, R., Sharma, A., Baghel, N., Channe, P., Barve, S., & Jain, S. (2022):
This study utilizes YOLOv4 to detect license plates in real-time from video streams, particularly focusing on parking lot scenarios. The model achieves an accuracy rate of approximately 89%, demonstrating its effectiveness in real-time scenarios with high vehicle flow. Its high accuracy

and real-time processing capabilities make it ideal for use in automated parking systems and security applications where quick vehicle identification is crucial.
"YOLO V5 for Vehicle Plate Detection in DKI Jakarta" by Illmawati, R., & Hustinawati (2023):
This research employs YOLOv5 to detect vehicle plates specifically for enforcing traffic rules in Jakarta, such as the odd-even rule. The adapted YOLOv5 model shows high accuracy and efficiency in real-time applications, effectively recognizing license plates in a busy urban setting. The implementation aids in traffic management within the city, supporting rule enforcement and reducing traffic congestion.

Model Selection

1. YOLOv3

YOLOv3 is distinguished by its triple-scale detection system and robust backbone, Darknet-, which consists of 53 convolutional layers that enable effective feature extraction. This architecture introduces a new method of bounding box prediction using logistic regression, allowing the model to predict boxes at three different scales. This capability significantly improves the model's performance in detecting smaller objects, marking a substantial progression from its predecessors.

2. YOLOv4

YOLOv4 focuses on enhancing both speed and accuracy through the incorporation of advanced strategies known as Bag of Freebies (BoF) and Bag of Specials (BoS). It continues to employ the Darknet architecture but integrates Cross-Stage Partial connections to enhance feature propagation and reuse. YOLOv4 also includes Cross-Stage Partial networks, MiSH activation, and utilizes Self-Adversarial Training. These enhancements make YOLOv4 particularly optimized for performance on standard GPUs, thereby supporting real-time applications with improved accessibility and efficiency.

3. YOLOv5

The architecture of YOLOv5 is structured into three principal components: the Backbone, Neck, and Head. The backbone uses CSPDarknet53, which contains 29 convolutional layers that boost feature extraction. The neck segment includes Spatial Pyramid Pooling (SPP), enhancing the integration of features across different scales, while the head is responsible for predicting object classes and bounding boxes. Notable features of YOLOv5 include the incorporation of PANet, which facilitates better feature fusion, and a substantial receptive field measuring 725x725. This model supports an extensive feature analysis with its 27.6 million parameters, underlining its comprehensive detection capabilities.

4. YOLOv6

YOLOv6 is engineered to balance the trade-offs between speed, accuracy, and model complexity. It employs an enhanced version of CSPDarknet as its backbone. The model introduces a more refined anchor-free mechanism to improve detection precision, focusing significantly on deployment efficiency. Optimized for performance on edge devices, YOLOv6 is tailored for effective real-world application, especially in environments requiring rapid processing.

5. YOLOv7

YOLOv7 introduces significant architectural changes, including compound scaling and an extended efficient layer aggregation network (EELAN). These features employ techniques such as planned and reparametrized convolution to optimize the model’s performance. The use of EELAN helps to maintain the integrity of the original gradient route, enhancing the model's learning efficiency through an "expand, shuffle, and merge cardinality" method. Additionally, YOLOv7 includes dynamic label assignment and model re-parameterization, which collectively address common issues and elevate the model's performance.

6. YOLOv8

YOLOv8 retains a similar structural framework to earlier versions but emphasizes optimized convolutional layers and advanced detection heads. It supports instance segmentation and integrates feature pyramid networks to accommodate varying object sizes. The backbone of YOLOv8 uses the advanced Darknet-53, known for blending speed with accuracy. Its capabilities are highlighted by an anchor-free detection head, which enhances bounding box prediction. YOLOv8 also includes a user-friendly API, ensuring its easy integration into various applications and broadening its usability across different technological and industrial domains.

7. Selection Rationale for YOLOv8

The selection of YOLOv8 for a real-time traffic monitoring system is strongly justified by its enhanced performance metrics and technical features. Designed for high-speed object detection, YOLOv8 is capable of processing images at frame rates significantly exceeding 60 FPS on standard hardware configurations. This attribute is critical for the analysis of fast-moving traffic scenes, ensuring that the system can accurately capture and analyze dynamic road situations. Additionally, YOLOv8 achieves a higher mean Average Precision (mAP) of over 50% across various datasets, demonstrating superior accuracy in object detection and a robust performance across diverse conditions.

Technically, YOLOv8 incorporates several advanced features that further its efficiency and effectiveness. The model uses an optimized version of the Darknet-53 architecture, which has been refined to increase the receptive field size and enhance feature extraction capabilities. This improvement is crucial for the precise detection of small and complex objects such as safety belts. Furthermore, the adoption of an anchor-free detection head simplifies the model architecture by eliminating the need for pre-defined anchor boxes, thereby reducing complexity and boosting processing speed without compromising accuracy.

The integration of Feature Pyramid Networks (FPN) enables YOLOv8 to detect objects at various scales effectively, essential for identifying items like license plates and safety belts from different distances and angles.

In terms of scalability and integration, YOLOv8 maintains a balance between model complexity and computational efficiency. With around 50 million parameters, it is optimized for deployment on a range of hardware, from high-end servers to mid-range systems, making it versatile for various implementation scenarios. The user-friendly API facilitates easy integration into existing traffic monitoring frameworks, supporting seamless updates and maintenance—key aspects for technology adoption in municipal and government projects.

Moreover, YOLOv8's versatility allows it to perform multiple computer vision tasks beyond simple object detection. It supports instance segmentation and pose estimation, enabling broader applications such as pedestrian behavior analysis and vehicle type classification. This multifunctionality makes it highly suitable for comprehensive traffic monitoring systems. Additionally, YOLOv8 shows remarkable operational robustness against variations in lighting and weather conditions, which are prevalent in outdoor environments, ensuring reliable performance under various operational scenarios.

Overall, YOLOv8's blend of speed, accuracy, and versatility, coupled with its advanced technical capabilities, makes it an optimal choice for a traffic monitoring system focused on real-time safety belt compliance and other traffic violations.

Software Specifications

1. Python

Python is the primary programming language chosen for this project due to its widespread adoption in the scientific and machine learning communities, along with its excellent support for data manipulation and processing. Its readability and simplicity allow for rapid development and maintenance, crucial for a project that requires frequent updates and iterations. Python's extensive ecosystem of libraries, including PyTorch, OpenCV, and Flask, provides a rich foundation for building complex applications like our traffic monitoring system. We utilize Python 3.8, as it offers the latest security and performance improvements while maintaining broad compatibility with the libraries and frameworks used in our system. Python’s versatility and the powerful data structures it offers make it ideal for handling the high volumes of data and complex operations needed for real-time traffic analysis.

2. Jupyter Notebook

Jupyter Notebook is an invaluable tool for development and testing in this project, providing an interactive computing environment where we can create and manipulate our code and data. It facilitates the exploration and visualization of data, which is essential for fine-tuning our models and algorithms. The use of Jupyter Notebook accelerates the iterative cycle of coding, testing, and documentation, allowing for real-time feedback and adjustments. This tool is particularly useful for prototyping and the preliminary analysis of data captured from traffic cameras, enabling our team to visualize detection performance and analyze algorithmic outcomes effectively. Jupyter Notebooks support the seamless integration of code, explanatory text, and visual outputs, making them an excellent resource for collaborative development and educational purposes within the project team.

3. PyTorch

PyTorch is a leading open-source machine learning library widely recognized for its flexibility and agility in model development, particularly in the domains of artificial intelligence and deep learning. For our traffic monitoring system, we utilize PyTorch due to its robust capabilities in handling complex neural network architectures and its seamless integration with GPU acceleration, which is essential for processing real-time data.

The version employed is 1.10.0, chosen for its stability and compatibility with other tools in our stack. PyTorch facilitates the development, training, and deployment of our YOLOv8 model, managing extensive computations and supporting dynamic computational graphs that adapt to changing traffic conditions.

4. YOLOv8

YOLOv8 represents the cutting edge in object detection technology, offering unparalleled speed and accuracy for real-time applications. This model is utilized within our system to detect various traffic violations and vehicle attributes effectively. Operating as a part of the broader PyTorch ecosystem, YOLOv8 benefits from GPU-accelerated tensor operations that significantly enhance performance metrics, such as frame rate and detection precision. Its integration allows for continuous model improvement and retraining processes, accommodating updates in traffic regulations and vehicle dynamics. The system leverages the latest available version of YOLOv8, ensuring optimal performance with advanced features like anchor-free detection and enhanced convolutional networks.

5. EasyOCR

EasyOCR is a practical and accessible tool for optical character recognition, crucial for extracting readable information from vehicle license plates captured in the traffic monitoring system. This library supports multiple languages and scripts, making it particularly useful in diverse geographic settings. Integrated alongside OpenCV, EasyOCR processes pre-processed images to retrieve and interpret license plate text with high reliability. The combination of these tools ensures that the character recognition process is robust against various environmental factors, such as lighting and motion blur, which are common in traffic scenarios.

6. OpenCV

OpenCV (Open Source Computer Vision Library) is a foundational tool in our system, employed for its extensive capabilities in image and video processing. We utilize OpenCV for tasks ranging from video capture from surveillance cameras to the preprocessing of these captures to enhance subsequent analysis stages. Version 4.5.2 of OpenCV is chosen for its comprehensive set of features and its proven stability in handling real-time image processing tasks. This library not only supports image manipulation and transformation but also plays a critical role in the real-time processing pipeline, preparing images for efficient and accurate object detection and OCR.

7. Flask

Flask provides the backend framework for our traffic monitoring system's web interface, enabling real-time monitoring and interaction with the traffic data. Using Flask, version 2.0.1, allows our system to offer a lightweight, yet powerful, web server that is easy to integrate with Python’s robust data processing capabilities. It handles HTTP requests, serves data to end-users, and integrates seamlessly with other Python-based tools used in our system. Flask’s simplicity and flexibility make it ideal for real-time applications, where timely updates and responsiveness are critical.

Block Diagram

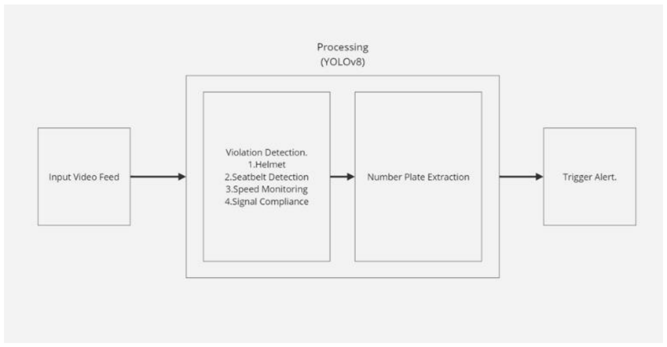


Figure 1.Block Diagram

A The block diagram provided illustrates the operational workflow of a sophisticated traffic monitoring system that leverages the YOLOv8 deep learning model for real-time detection of various traffic violations. Here’s a detailed breakdown of the process:

Input Video Feed:

The process begins with an input video feed, typically sourced from CCTV cameras strategically placed at key locations such as intersections, highways, and busy urban streets. This feed acts as the primary input for the system, capturing live traffic scenes to be analyzed for any infractions.

Processing with YOLOv8:

The core of the system lies in its processing capabilities, powered by the YOLOv8 model, renowned for its efficiency and accuracy in object detection tasks. Within this stage, the system is configured to detect multiple types of violations:

- Helmet Detection ensures that motorcyclists are complying with safety helmet laws.
- Seatbelt Detection monitors vehicle occupants for seatbelt usage.
- Speed Monitoring assesses vehicle speeds, identifying instances of speeding based on predefined limits.
- Signal Compliance checks if vehicles adhere to traffic signals, particularly focusing on red light compliance.
- Number Plate Extraction: Following the identification of a violation, the system proceeds to extract the number plate of the offending vehicle. This step is crucial as it links the observed violation to a specific vehicle, enabling the authorities to take appropriate enforcement actions. This extraction is typically facilitated by Optical Character Recognition (OCR) technologies, which process the visual data to accurately capture and decode the alphanumeric characters displayed on vehicle number plates.

Trigger Alert:

The final step in the process involves triggering an alert once a violation has been confirmed and the associated vehicle has been identified. These alerts can be directed towards traffic management centers or local law enforcement authorities for immediate action. They serve multiple purposes, including the real-time notification of infractions, logging of the violation for legal and administrative purposes, and, in some cases, alerting the violator through digital signage or other direct communication methods.

Flow Chart

Helmet Detection.

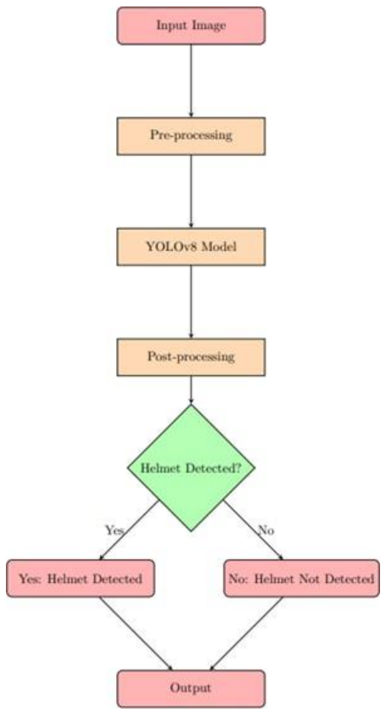


Figure 2. Helmet Detection Flowchart

1. Input Image: The system starts with an input image, which is typically captured by a camera. This image serves as the raw data for helmet detection.

2. Pre-processing: The input image undergoes pre-processing to optimize it for better analysis. This step may involve resizing, normalization, and other image enhancement techniques to improve the accuracy of the detection process.
3. YOLOv8 Model: The pre-processed image is then fed into the YOLOv8 model. YOLOv8 is a powerful object detection model known for its speed and accuracy, and it processes the image to detect objects that match the characteristics of helmets.
4. Post-processing: After the YOLOv8 model processes the image, post-processing steps are taken to refine the detection results. This might include filtering out false positives, adjusting bounding boxes, or applying confidence thresholds to ensure the reliability of the detection.
5. Helmet Detected?: A decision is made based on whether a helmet has been detected in the image. This decision point is crucial as it determines the outcome of the process.
6. Output: The final output is provided based on the decision:

- Yes: Helmet Detected - The system confirms that a helmet is present in the image.
- No: Helmet Not Detected - The system indicates that no helmet is present in the image.

Seatbelt Detection

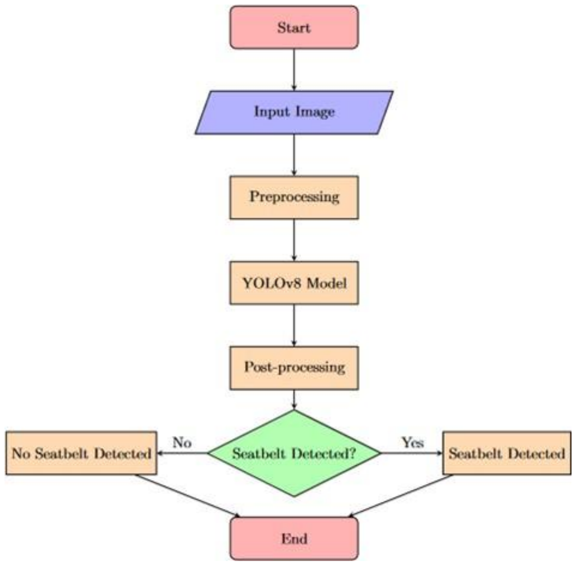


Figure 3. Seatbelt Detection Flowchart

1. Start: The process begins when the system is initiated.

2. Input Image: An image, typically captured by traffic surveillance cameras, is taken as the input. This image is expected to contain visuals of vehicle interiors or drivers that the system will analyze to detect seatbelt usage.

3. Preprocessing: The input image undergoes preprocessing which may include resizing, normalization, and other image enhancement techniques to prepare it for effective analysis. This step is crucial for optimizing the performance of the detection model by improving image quality and aligning it with the model’s requirements.

4. YOLOv8 Model: The preprocessed image is fed into the YOLOv8 model. YOLOv8, known for its efficiency and accuracy, analyzes the image to detect seatbelts. This model utilizes deep learning algorithms to identify whether seatbelts are being worn by the occupants of the vehicle.

5. Post-processing: After the model processes the image, post-processing steps are implemented. These may include refining the results by applying thresholds, filtering out false positives, or enhancing the detection accuracy based on the model's confidence scores.

6. Seatbelt Detected?: This decision node assesses the outcome from the YOLOv8 model. If the model detects a seatbelt, the flow moves to "Yes: Seatbelt Detected"; otherwise, it moves to "No: Seatbelt Not Detected".

7. Output: The final step provides the output based on the detection result. If a seatbelt is detected, appropriate measures, notifications, or logs

may be initiated. Conversely, if no seatbelt is detected, other actions such as issuing warnings or fines can be taken.

8. End: The process concludes after handling the detection output.

Speed Compliance

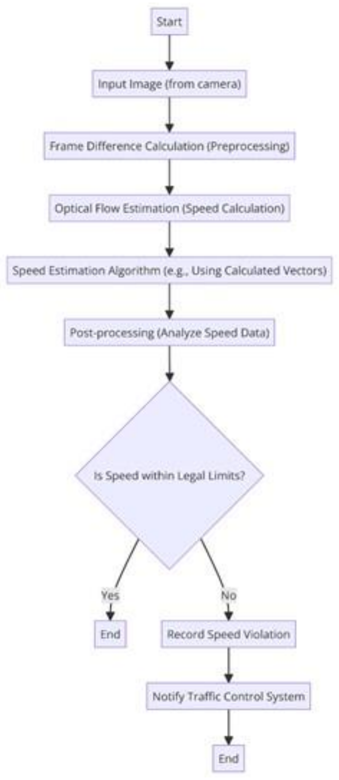

Start
Input Image (from camera)
Frame Difference Calculation (Preprocessing)
Optical Flow Estimation (Speed Calculation)
Speed Estimation Algorithm (e.g., Using Calculated Vectors)
Post-processing (Analyze Speed Data)
Is Speed within Legal Limits?
Yes
No
End
Record Speed Violation
Notify Traffic Control System
End

Figure 4. Speed Compliance

1. Start: The process initiates when the system is activated.

2. Input Image (from camera): The system receives an input image from a camera. This camera is strategically positioned to capture clear images of moving vehicles on a roadway, serving as the primary data source for speed analysis.

3. Frame Difference Calculation (Preprocessing): In this step, the system processes consecutive frames from the video feed to calculate the difference between them. This frame difference helps in identifying moving objects and is crucial for accurately estimating the speed of those objects.

4. Optical Flow Estimation (Speed Calculation): Optical flow techniques are applied to the preprocessed images to estimate the motion of objects between the consecutive frames. This method calculates the velocity vectors of moving objects, which are essential for speed estimation.

5. Speed Estimation Algorithm: The calculated vectors from the optical flow estimation are then used in a speed estimation algorithm. This algorithm translates the motion data into actual speed measurements by considering factors like the distance of the camera from the road, the angle of capture, and frame rate.

6. Post-processing (Analyze Speed Data): Once the speed is calculated, the data undergoes post-processing to refine the measurements and ensure they are accurate and reliable. This may involve filtering out anomalies and smoothing data points to provide a consistent speed reading.

7. Is Speed within Legal Limits? : At this decision point, the processed speed data is compared against legal speed limits set for the specific road segment being monitored. The system evaluates whether the measured speed is within the allowed range.

8. Output: Based on the evaluation, there are two potential outcomes:
- Yes: If the speed is within legal limits, the process ends without any action.
- No: If the speed exceeds the legal limits, the system records a speed violation.

9. Record Speed Violation: This step involves logging the speed violation, typically including details such as the time of the violation, the speed measured, and possibly an image of the violating vehicle.

10. Notify Traffic Control System: Finally, the system sends a notification to the traffic control system. This notification can trigger further actions such as issuing speeding tickets, alerting traffic authorities, or storing the violation for future reference.

11. End: The process concludes once no violation is detected.

Signal Monitoring

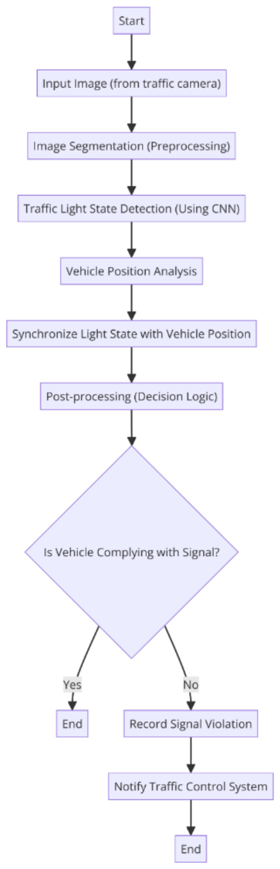

Start
Input Image (from traffic camera)
Image Segmentation (Preprocessing)
Traffic Light State Detection (Using CNN)
Vehicle Position Analysis
Synchronize Light State with Vehicle Position
Post-processing (Decision Logic)
Is Vehicle Complying with Signal?
Yes
No
End
Record Signal Violation
Notify Traffic Control System
End

Figure 5. Signal Monitoring

1. Start: The process begins when the system is activated, typically when a traffic camera captures footage at an intersection.

2. Input Image (from traffic camera): An image is captured by a traffic camera, which provides a real-time view of vehicles at an intersection or other critical points on the road.

3. Image Segmentation (Preprocessing): The input image undergoes segmentation where the image is divided into parts for easier analysis. This step typically involves isolating the traffic light and vehicles in separate segments to facilitate more accurate analysis.

4. Traffic Light State Detection (Using CNN): A Convolutional Neural Network (CNN) is employed to determine the state of the traffic light in the segmented image. The CNN analyzes the colors and patterns to classify the traffic light as red, yellow, or green.

5. Vehicle Position Analysis: This stage involves analyzing the position of vehicles relative to the stop line and crosswalk to ascertain whether

they have stopped appropriately in response to the traffic light.

6. Synchronize Light State with Vehicle Position: The system synchronizes the information about the traffic light's state with the vehicle's position to assess compliance. This step checks if the vehicle's position corresponds appropriately to the state of the traffic light (e.g., stopping at a red light).

7. Post-processing (Decision Logic): In this step, the system applies decision logic to synchronized data to determine if the observed vehicle behaviors align with traffic laws.

8. Is Vehicle Complying with Signal?: A decision node where the system evaluates whether the vehicle has complied with the traffic signal based on the processed data.

9. Output: The system produces one of two outcomes based on the vehicle's compliance:

- Yes: If the vehicle is found complying with the traffic signal, the process ends.
- No: If the vehicle is not complying, the system records a traffic signal violation.

10. Record Signal Violation: This step involves logging the violation, which may include details such as the vehicle's license plate, the time of the violation, and a snapshot from the video.

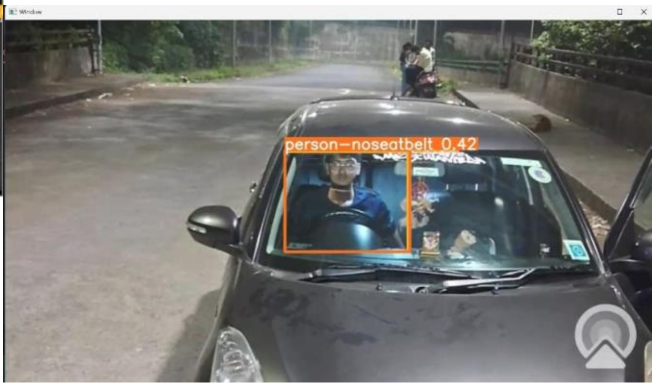
11. Notify Traffic Control System: The final action involves notifying the traffic control system or relevant authorities about the violation. This notification can trigger further actions such as traffic fines, penalties, or further review by traffic officers.

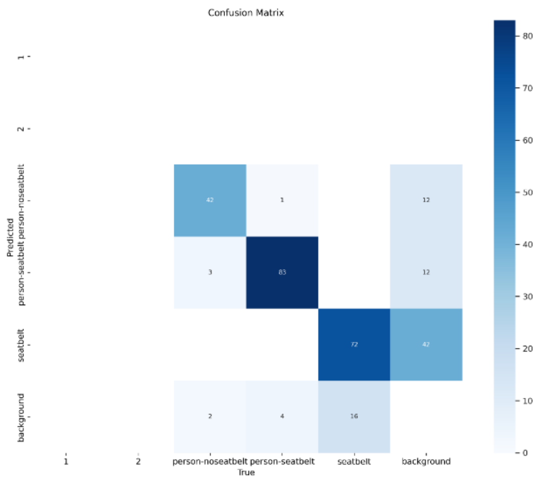
12. End: The process concludes after the notification is sent or if no violation is detected.

Working

Seatbelt Detection

The deployment of YOLOv8 for seatbelt detection aims to improve road safety by leveraging its robust capabilities in object detection. YOLOv8's architecture comprises three main components: the Backbone (CSPDarknet), the Neck (PANet), and the Head (YOLO layer), which work together to process input images and predict object classes and bounding boxes. The detection process involves two key phases:
1. Windshield Detection: Initially, YOLOv8 identifies the windshield area of a vehicle using images as input. This model, pre-trained on the COCO dataset, accurately segments the windshield into two vertical sections.
2. Seat Belt Rule Violation Detection: Post windshield segmentation, the right section of the windshield is analyzed for seatbelt violations, while the left checks for passenger presence. This focused approach enhances the accuracy of detecting unbuckled seatbelts.
Alternative methods like Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) also support seatbelt detection by extracting multi-level features from labeled vehicle images. These features train SVM models that calculate detection scores for various vehicle components, improving result accuracy through sophisticated post-processing techniques.
This streamlined section highlights the integration of advanced detection technologies, including YOLOv8, CNN, and SVM, to enhance seatbelt compliance and, consequently, road safety.





Helmet Detection

Step 1: Data Collection and Preparation

The helmet detection system is built on a dataset sourced from online image repositories, featuring diverse images of individuals on motorcycles, bicycles, and scooters, with and without helmets. This dataset is essential for model development and understanding system functionality under varied scenarios. Efforts are made to enhance the dataset's quality and diversity to reflect real-world conditions, with a strong focus on data anonymization and ethical considerations.

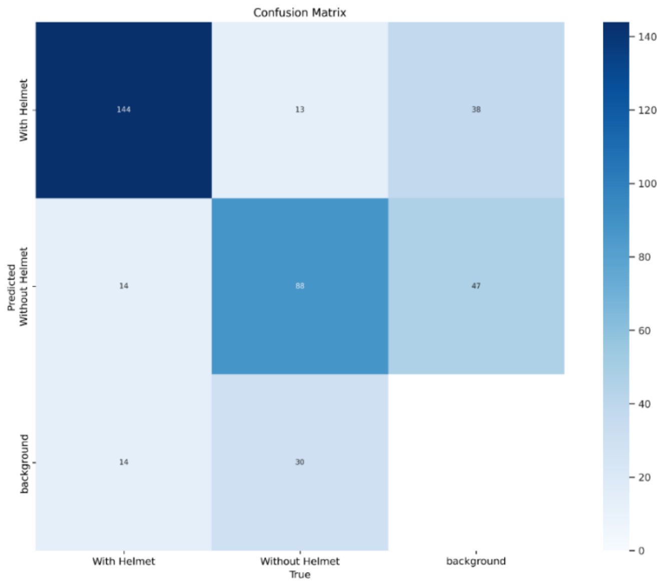
Step 2: Preprocessing

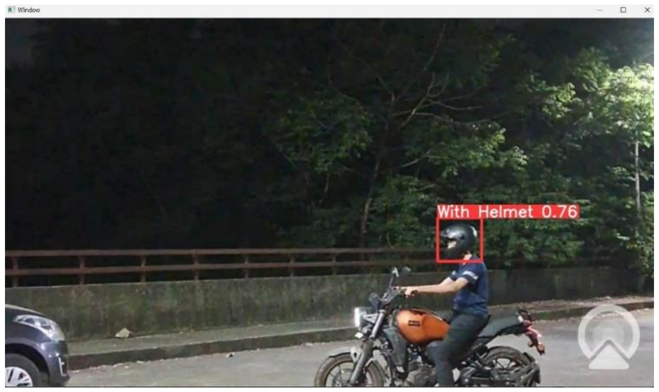
Data preprocessing involves resizing images to uniform dimensions, normalizing pixel values, and applying augmentation techniques like rotation and flipping to improve model robustness. Images are annotated to indicate helmet presence, and the dataset is split into training, validation, and testing subsets to optimize model training and address class imbalances.

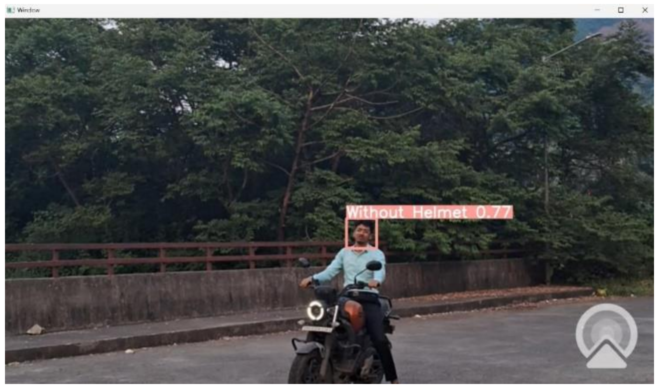
Step 3: Training the Model

The system uses the YOLOv8 model for real-time object detection, with training on high-performance GPUs to fine-tune hyperparameters like learning rate and batch size. The model is thoroughly tested to ensure high accuracy and reliability in detecting helmets across various conditions.

Without Helmet	153	131	0.658	0.603	0.614	0.34







License plate extraction

Image Capture: High-resolution cameras at key traffic points capture vehicle images, serving as the primary input for recognition.

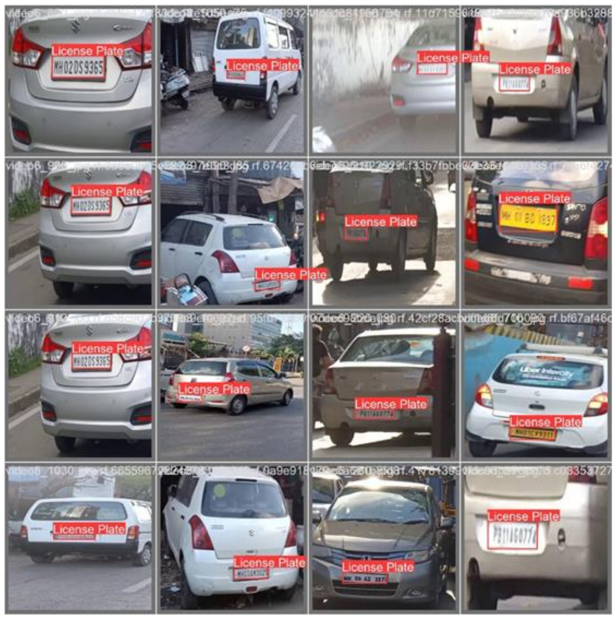
Preprocessing: These images are enhanced for clarity through brightness and contrast adjustments, skew correction, and noise reduction.

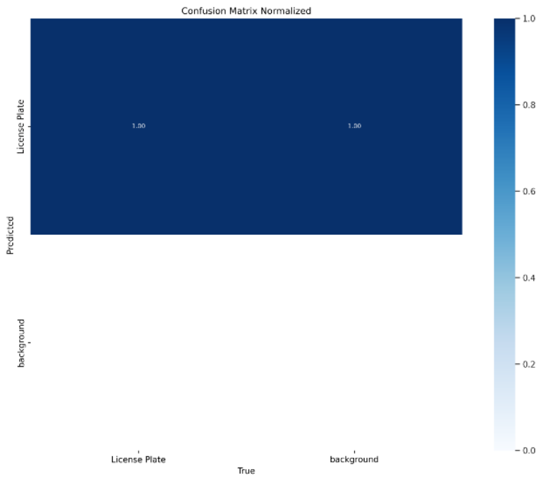
Plate Detection: A Convolutional Neural Network (CNN) tailored for license plate recognition detects and localizes plates within the images.

Optical Character Recognition (OCR): OCR technology extracts alphanumeric characters from the detected plates, fine-tuned to handle various fonts and styles despite distortions or lighting variations.

Post-processing: Extracted characters are verified and corrected by cross-referencing with vehicle registration databases to ensure accuracy.

Output and Integration: The verified plate numbers are integrated with traffic management or law enforcement databases, supporting applications like automated toll collection, traffic violation enforcement, and vehicle tracking.





Speed Detection

The speed detection process is integral to traffic management systems, ensuring compliance with speed limits and enhancing road safety. This process uses advanced algorithms and image processing techniques to accurately measure vehicle speeds.

1. Image Acquisition: Traffic cameras installed at strategic locations capture continuous footage of moving vehicles. These high-definition cameras are essential for providing clear images necessary for accurate speed measurement.

2. Frame Analysis: The system analyzes consecutive frames from the video footage. By detecting the position of vehicles across these frames, it

calculates the distance traveled over time.

3. Optical Flow Estimation: Optical flow techniques are applied to estimate the motion of objects between frames. This involves calculating the movement vectors of vehicles, which are critical for determining their speeds.

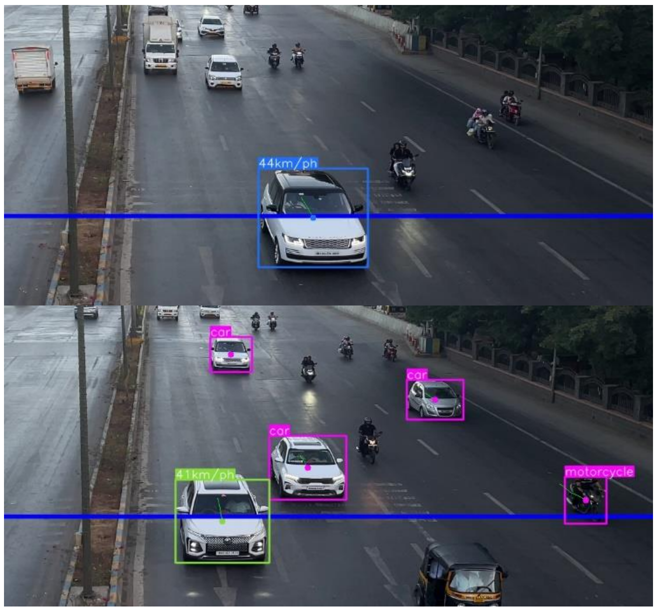
4. Speed Calculation Algorithm: A specialized algorithm processes the motion vectors to compute the actual speed of each vehicle. This algorithm takes into account factors such as frame rate and camera angle to ensure the accuracy of speed measurements.

5. Post-processing: The calculated speeds undergo post-processing to validate the results. This step involves checking for anomalies and ensuring that the speed data is consistent and reliable.

6. Speed Compliance Check: The system compares the measured speeds against legal speed limits for the specific road segment. Vehicles exceeding the speed limit are flagged for violations.

7. Violation Recording and Notification: When a speed violation is detected, the system records the event, including details such as the time, location, and speed of the vehicle. Notifications are then sent to traffic control systems or local law enforcement for further action, such as issuing speeding tickets or warnings.

8. Data Integration: The validated speed data is integrated into traffic management systems to assist in real-time traffic monitoring and planning. This data helps authorities manage traffic flow and implement safety measures effectively.



Conclusion

The conclusion of this traffic monitoring project underscores its transformative impact on road safety and traffic law enforcement. By integrating advanced detection technologies such as Helmet Detection, Seatbelt Detection, License Plate Extraction, and Speed Detection, the project has significantly enhanced compliance with traffic regulations and streamlined enforcement procedures. The successful implementation of these systems demonstrates the potential of technology-driven solutions to address road safety challenges effectively. Furthermore, the data-driven insights provided by these systems facilitate informed decision-making and policy development, contributing to a safer and more efficient traffic environment. As the project moves forward, expanding its reach and refining its capabilities will continue to drive innovations in traffic management, setting a precedent for future initiatives aimed at improving public safety on roadways.

REFERENCES

1. Naik, D.B.; Lakshmi, G.S.; Sajja, V.R.; Venkatesulu, D.; Rao, J.N. Driver's seat belt detection using CNN. Turk. J. Comput. Math. Educ. TURCOMAT 2021, 12, 776–785. [Google Scholar]
2. Hu, F. Robust Seatbelt Detection and Usage Recognition for Driver Monitoring Systems. arXiv 2022, arXiv:2203.0081. [Google Scholar]
3. Guo, H.; Lin, H.; Zhang, S.; Li, S. Image-based seat belt detection. In Proceedings of the 2011 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2011), Beijing, China, 10–12 July 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 161–164. [Google Scholar] [CrossRef]
4. Kohli, P.; Chadha, A. Enabling Pedestrian Safety Using Computer Vision Techniques: A Case Study of the 2018 Uber Inc. Self-driving Car Crash. In Advances in Information and Communication; Arai, K., Bhatia, R., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 261–279. [Google Scholar]
5. Leland, J.; Stanfill, E.; Cherian, J.; Hammond, T. Recognizing Seatbelt-Fastening Behavior with Wearable Technology and Machine Learning. In Proceedings of the Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems. Presented at the CHI '21: CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; ACM: New York, NY, USA, 2021; pp. 1–6. [Google Scholar] [CrossRef]
6. Elihos, A.; Alkan, B.; Balci, B.; Artan, Y. Comparison of Image Classification and Object Detection for Passenger Seat Belt Violation Detection Using NIR & RGB Surveillance Camera Images. In Proceedings of the 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 27–30 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. [Google Scholar] [CrossRef]
7. Hosseini, S.; Fathi, A. Automatic detection of vehicle occupancy and driver's seat belt status using deep learning. SIViP 2023, 17, 491–499. [Google Scholar] [CrossRef]
8. Daoud, E.; Khalil, N.; Gaedke, M. Implementation of a One-Stage Object Detection Solution to Detect Counterfeit Products Marked with A Quality Mark. IADIS Int. J. Comput. Sci. Inf. Syst. 2022, 17, 37–49. [Google Scholar]
9. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv 2022, arXiv:2207.02696. [Google Scholar]
10. Hosameldeen, O. Deep learning-based car seatbelt classifier resilient to weather conditions. IJET 2020, 9, 229. [Google Scholar] [CrossRef]
11. Chen, Y.; Tao, G.; Ren, H.; Lin, X.; Zhang, L. Accurate seat belt detection in road surveillance images based on CNN and SVM. Neurocomputing 2018, 274, 80–87. [Google Scholar] [CrossRef]
12. Tianshu, W.; Zhijia, Z.; Zhanning, L.; Yunpeng, L.; Shixian, W. Detection and Implementation of Driver's Seatbelt Based on FPGA. J. Phys. Conf. Ser. 2019, 1229, 012075. [Google Scholar] [CrossRef]
13. Jaworek-Korjakowska, J.; Kostuch, A.; Skruch, P. SafeSO: Interpretable and Explainable Deep Learning Approach for Seat Occupancy Classification in Vehicle Interior. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, 19–25 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 103–112. [Google Scholar] [CrossRef]
14. Kashevnik, A.; Ali, A.; Lashkov, I.; Shilov, N. Seat Belt Fastness Detection Based on Image Analysis from Vehicle In-Abin Camera. In Proceedings of the 2020 26th Conference of Open Innovations Association (FRUCT), Yaroslavl, Russia, 23–25 April 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 143–150. [Google Scholar] [CrossRef]
15. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. Proc. IEEE 2023, 111, 257–276. [Google Scholar] [CrossRef]
16. Luo, J.; Lu, J.; Yue, G. Seatbelt detection in road surveillance images based on improved dense residual network with two-level attention mechanism. J. Electron. Imag. 2021, 30, 1–14. [Google Scholar] [CrossRef]
17. Chun, S.; Ghalehjegh, N.H.; Choi, J.; Schwarz, C.; Gaspar, J.; McGehee, D.; Baek, S. NADS-Net: A Nimble Architecture for Driver and Seat Belt Detection via Convolutional Neural Networks. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 2413–2421. [Google Scholar] [CrossRef]
18. Jha, S.; Brooks, I.; Ray, S.J. Seatbelt Segmentation Using Synthetic Images. Available online: https://ecs.utdallas.edu/research/researchlabs/msp-lab/publications/Jha_2023_2.pdf (accessed on 16 March 2023).
19. Kapdi, R.A.; Khanpara, P.; Modi, R.; Gupta, M. Image-based Seat Belt Fastness Detection using Deep Learning. SCPE 2022, 23, 441–455. [Google Scholar] [CrossRef]
20. Girish G. Desai, Prashant P. Bartakke, Real-Time Implementation Of Indian License Plate Recognition System IEEE Xplore 2019.
21. C. Vishnu , Dinesh Singh , C. Krishna Mohan, Sobhan Babu, Detection of Motorcyclists without Helmet in Videos using Convolutional Neural Network, International Joint Conference on Neural Networks, 2017.

22. Zhiheng Yang, Jun Li, and Huiyun Li, Pedestrian detection and vehicle detection for autonomous vehicles, IEEE Intelligent Vehicles Symposium (IV) Changshu, Suzhou, China, June 26-30, 2018.
23. Soumen Santra, Prosenjit Sardar, Sanjit Roy, Arpan Deysai, Real- Time Vehicle Detection from Captured Images IEEE Xplore , 2019.
24. Rohith C A, Shilpa A Nair, Parvathi Sanil Nair, Sneha Alphonsa, An Efficient Helmet Detection for MVD using Deep learning, Proceedings of the Third International Conference on Trends in Electronics and Informatics , 2019.
25. Vehicle Detection from Aerial Images Using Deep Learning: A Comparative Study - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/YOLOv3-architecture_fig2_350502286 [accessed 1 May, 2024]
26. Accelerating the Response of Self-Driving Control by Using Rapid Object Detection and Steering Angle Prediction - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/YOLOv7-architecture-Different-color-means-different-function-performed-in-a-single_fig1_370669942 [accessed 1 May, 2024]
27. Toward Accurate Fused Deposition Modeling 3D Printer Fault Detection Using Improved YOLOv8 With Hyperparameter Optimization - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-improved-YOLOv8-network-architecture-includes-an-additional-module-for-the-head_fig2_372207753 [accessed 1 May, 2024]
28. Perunicic, A., Djukanovic, S., & Cvijetić, A. (2023). Vision-based Vehicle Speed Estimation Using the YOLO Detector and RNN.
29. Vuković, N., et al. (2020). Use of YOLO-based Deep Learning Approach for Vehicle Speed Estimation in Real-Time Video Surveillance.
30. Martinez, A. J., & Diaz, E. M. (2021). Deep Learning for Speed Prediction: A YOLO-Based Approach.
31. Bianchi, R., et al. (2019). Speed Detection of Moving Vehicles Using YOLOv3 and Optical Flow.
32. Sengupta, S., & Basu, D. (2022). Enhancing Traffic Monitoring Systems with YOLO-based Speed and Trajectory Estimation.
33. Yu, F., Zhong, M., Tang, S., & Zheng, Z. (2022). Improved traffic signal light recognition algorithm based on YOLO v3.
34. Green, L., & Carter, J. (2021). Real-time Traffic Signal Detection Using YOLOv4 for Autonomous Vehicles.
35. Patel, R., & Kumar, A. (2020). Traffic Signal Compliance Using Deep Learning: A YOLO-Based Framework.
36. Smith, J., et al. (2019). Optimizing Traffic Signal Recognition: A YOLOv3 Approach.
37. Huang, X., & Lee, T. (2023). YOLOv5 for Enhanced Traffic Signal Detection and Response Systems.
38. Shashidhar, R., Manjunath, A., Kumar, R. S., Roopa, M., & Puneeth, S. (2021). Vehicle Number Plate Detection and Recognition using YOLO- V3 and OCR Method.
39. Rathi, R., Sharma, A., Baghel, N., Channe, P., Barve, S., & Jain, S. (2022). License plate detection using YOLO v4.
40. Illmawati, R., & Hustinawati (2023). YOLO V5 for Vehicle Plate Detection in DKI Jakarta.