AI-Driven Traffic Dynamics: Enhancing Flow and Reducing Congestion: Review

1st Prof. Sagar B. Shinde
Head Of Department CSE-AI,
Nutan Maharashtra Institute of Engineering and Technology, Talegaon
Pune, India
sagar.shinde5736@gmail.com

2nd Mr. Devarth B. Patel
Department of CSE-AI,
Nutan College of Engineering and Research , Talegaon
Pune, India
devarthpatel128@gmail.com

3rd Ms. Rashmi Murugkar
Department of CSE-AI,
Nutan College of Engineering and Research , Talegaon
Pune, India
rashmirm99@gmail.com

4th Ms. Shruti Jadhav
Department of CSE-AI,
Nutan College of Engineering and Research , Talegaon
Pune, India
shrutijadhav1863@gmail.com

5th Mr. Prathamesh Anandaraj
Department of CSE-AI,
Nutan College of Engineering and Research , Talegaon
Pune, India
prathamesh1402@gmail.com

Abstract— Traffic congestion at multi-lane intersections in urban areas is a critical challenge especially in large metropolitan cities such as New Delhi, Mumbai, Bengaluru and many more. Traditional traffic signals with fixed timers has become outdated and can't adapt to real-time traffic conditions which has led to increased delays and more fuel consumption. This model aims to address issue by implementing machine learning-based adaptive signal control system by using YOLO (You Only Look Once) which is employed for real-time vehicle detection and counting across multiple traffic lanes using CCTV images. Based on vehicle density in each lane, dynamic green signal timings are assigned to optimize traffic flow, with higher priority given to congested lanes. A virtual red line is drawn in each lane, and vehicles crossing this line during a red signal are automatically flagged and marked as violators. By integrating these technologies, the system promises to improve traffic flow and reduce congestion saving a lot of precious time for the commuters.

Key Words - Traffic congestion, Signal Timers.

I. Introduction

The challenge of managing urban intersections has grown significantly alongside the rapid expansion of cities and the increasing number of vehicles on roads [7]. Urban traffic congestion is no longer just a logistical concern but a critical issue affecting safety, commute efficiency, and urban planning. Traditional traffic control systems rely heavily on pre-set signal timings [1], which are poorly suited for dynamic and unpredictable traffic conditions. As a result, such systems often lead to longer wait times, increased fuel consumption, and elevated risks of traffic violations and accidents.

Modern cities require smarter, data-driven systems capable of adapting in real-time to fluctuating traffic patterns [2]. This has driven research interest toward integrating computer vision and machine learning techniques into traffic management. These technologies can analyze real-time data, detect patterns, and adjust signal timings dynamically, offering both efficiency and safety benefits at congested intersections [3].

One promising direction involves real-time traffic monitoring using video surveillance combined with intelligent detection models. For example, Silva and Jung (2021) presented a flexible computer vision-based method aimed at improving vehicle detection in real-time scenarios [8]. Their system highlighted the importance of high-quality visual data for accurate traffic monitoring, although it struggled under adverse weather conditions. Similarly, another study introduced a dynamic graph neural network model to estimate traffic density [5]. Despite its potential, it faced challenges in processing large datasets, limiting its scalability for major metropolitan networks.

To address such limitations, this project proposes a smart traffic control system that combines the robustness of computer vision with the adaptability of machine learning [4]. The goal is to build a framework that can monitor vehicle density at intersections, even under challenging environmental conditions. By using visual input from traffic cameras [11], the system can identify and count vehicles in real-time. This count can then be used to optimize signal timings dynamically—minimizing idle time at signals and improving overall traffic throughput [12].

Several recent studies have contributed valuable insights into the field. For instance, research has explored the use of real-time anomaly detection for urban traffic management, focusing on unpredictable traffic surges and irregularities [3]. Although not directly linked to traffic light control, the findings underscore the value of responsive systems that adjust to real-world conditions. Another study utilized queuing theory to develop cost-effective traffic models [4], though its reliance on simulated data limits its real-world impact.

The importance of scalable and adaptable traffic systems is further emphasized in [6], which proposed using sensor networks to detect anomalies in urban traffic flow. Meanwhile, [7] reviewed neural network-based detection systems and noted the challenges of requiring large annotated datasets for effective training—highlighting the trade-off between model performance and data availability.

Further innovation comes from [8], which optimized a YOLOv8-based model to detect safety violations and traffic patterns. Though the model demonstrated improved responsiveness, it also faced slight accuracy reductions under certain scenarios. This suggests a broader need for systems that maintain reliability across diverse conditions without being overly dependent on ideal inputs [9].

Recent advancements in traffic forecasting have also leaned on sophisticated models such as Graph Multi-Attention Networks (GMAN) [6], which use attention mechanisms to better understand complex urban traffic patterns. MRA-BGCN [5], another novel model, enhances prediction accuracy by integrating node and edge-level interactions, improving the understanding of localized and interconnected traffic flows.

In [11], researchers analyzed traffic congestion through various indicators—such as travel time, speed, and volume-to-capacity ratios—offering practical ways to measure urban mobility and diagnose problem zones. These congestion metrics are essential for evaluating the effectiveness of any traffic control intervention [13].

Lastly, the application of deep learning models for real-time traffic environment analysis has also gained traction. A system utilizing YOLO for object detection and SORT for vehicle tracking demonstrated how real-time visual data can inform signal control decisions, improving response to traffic conditions as they evolve [11][13]. Additional research highlighted the benefits of AI-driven traffic signal systems and sustainable traffic management strategies [14][15][16].

More recent developments have seen improved algorithms for detecting traffic congestion and advanced YOLO-based models tailored for real-time urban environments [17][18][19].

In conclusion, managing urban intersections demands intelligent, adaptable systems. This project aims to contribute by leveraging computer vision and machine learning to create a dynamic, real-time traffic control framework. By focusing on real-time density monitoring and visual analysis, the proposed system can enhance signal efficiency, improve traffic flow, and support safer, more organized urban mobility—without depending on enforcement-based methods.

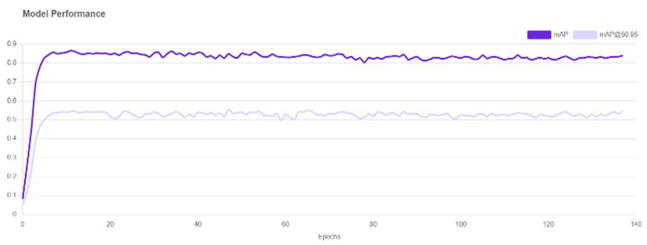


Fig 1. YOLOv8 Model Performance Graph Over Epochs (mAP and mAP@50:95)

Table I: Comparison table between YOLO-v8 NAS object detection and Roboflow 3.0 object detection

ASPECT	**YOLO-V8**	**ROBOFLOW 3.0**
Accuracy (Detection Quality)	Good dataset and annotations makes the model robust to some noise variations.	Sometimes less consistent in edge cases if used with same datasets.
Integration	Seamless with Ultralytics' YOLO class, as seen in our vehicle_model = YOLO(...) line.	Requires custom code depending on export format (.pb, ONNX, TF, TFLite, etc.)
Performance (Speed)	Highly optimized and fast with real-time datasets.	Varies by chosen model during Roboflow export; may be slower depending on backend
Training flexibility	CLI/SDK training, hyperparameter tuning, scripting possible.	Web UI-based training: tuning is limited unless retrained offline or downloaded.
Code reusability	Code written is already optimized by Yolo.	Needs partial rewrite to support another inference structure(ONNX/TFLite/etc.)
Visualization	Easy integration with **matplotlib** as used in our current workflow.	Parsing detection output might need extra steps before drawing.

II. METHODOLOGY OF PROPOSED SYSTEM

The proposed system is designed to enhance urban traffic management by combining computer vision, deep learning, and reinforcement learning to dynamically control traffic signals and detect red-light violations. The process begins with continuous traffic monitoring through CCTV cameras installed at road junctions, which capture real-time video feeds of traffic flow. These live feeds serve as the raw input for the system. The video is then processed in real-time using a deep learning model—YOLOv8—for vehicle detection and tracking. YOLOv8 efficiently identifies different types of vehicles such as cars, bikes, buses, and trucks in each frame, providing accurate vehicle count and positional data. This structured vehicle data is sent to the traffic data collection module, where it is aggregated from all lanes to create a comprehensive traffic snapshot at the intersection. The system uses this data to determine the density of each lane and identifies vehicle positions relative to a virtual red line that acts as a zebra crossing for red-light violation detection.

The collected traffic data is then used by the adaptive signal scheduling module, which employs either a basic Round Robin algorithm or an advanced Reinforcement Learning (RL) model. While Round Robin gives equal timing to all lanes in sequence, the RL model intelligently adjusts the duration of green lights based on current traffic density, aiming to reduce congestion and minimize waiting times. The RL model learns from both real-time and historical data to improve decision-making over time. In parallel, the red line logic monitors vehicles that cross the line during a red signal and flags them as violators, allowing authorities to take necessary action. Based on the optimized signal timings generated by the scheduler, the system controls the traffic signal lights, giving priority to heavily congested lanes and improving overall traffic flow.

The final module, Evaluation and Optimization, plays a critical role in assessing the performance of the system. It evaluates parameters such as traffic flow efficiency, number of red-light violations, average waiting time per vehicle, and the effectiveness of the adaptive signal scheduling approach. These insights are fed back into the system to further fine-tune the RL model and improve its accuracy. Over time, this feedback loop helps the system adapt to new traffic behaviors and maintain optimal performance. By integrating computer vision, AI, and real-time analytics, the system offers a scalable and intelligent solution for traffic signal management and law enforcement in modern cities.

III. ALGORITHM AND IMPLEMENTATION

Python is the core programming language used to develop the entire intelligent traffic management system. Its simplicity, readability, and extensive library ecosystem make it ideal for integrating various components such as image processing, deep learning models, and reinforcement learning algorithms. Python serves as the foundation for controlling logic flow, handling data manipulation, and interfacing with computer vision and machine learning libraries.

OpenCV (Open Source Computer Vision Library) is used for image processing and visualization tasks within the project. It enables the system to read, modify, and annotate CCTV frames for tasks like drawing lane dividers, placing virtual red lines for violation detection, and visualizing bounding boxes around detected vehicles. OpenCV plays a crucial role in pre-processing images before they are passed into the YOLO model and in providing visual feedback to developers and enforcement teams.

YOLO (You Only Look Once) by Ultralytics is the deep learning model utilized for real-time vehicle detection. YOLOv8, known for its high accuracy and speed, can detect multiple vehicles in a single frame and classify them by type (e.g., car, bike, bus). It processes entire images at once rather than scanning through regions, making it exceptionally fast and suitable for real-time traffic monitoring applications where timely detection is critical.

PyTorch is a popular open-source machine learning framework used for implementing and deploying the reinforcement learning model in the system. In this context,

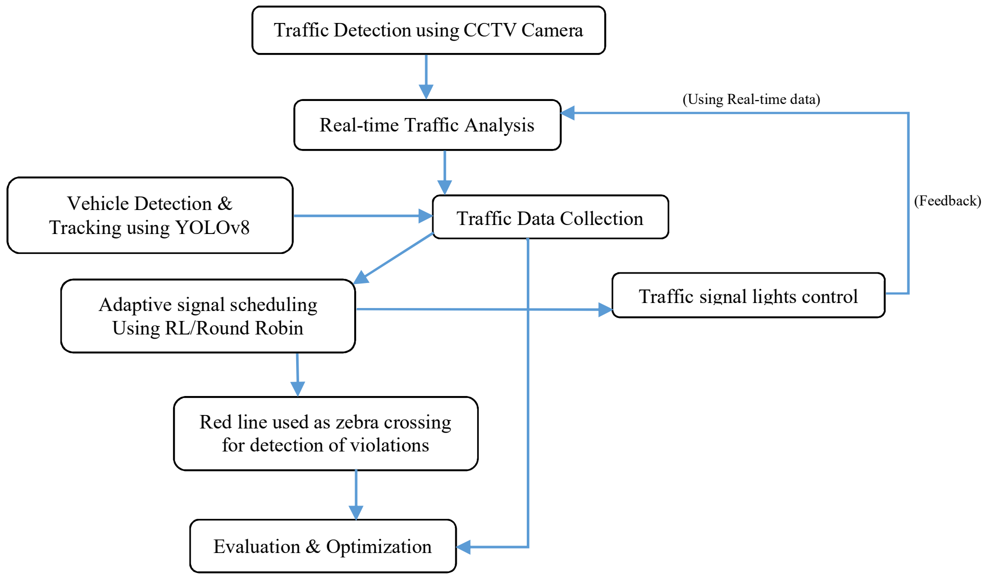

Traffic Detection using CCTV Camera
Real-time Traffic Analysis
(Using Real-time data)
(Feedback)
Vehicle Detection & Tracking using YOLOv8
Traffic Data Collection
Adaptive signal scheduling Using RL/Round Robin
Traffic signal lights control
Red line used as zebra crossing for detection of violations
Evaluation & Optimization

Fig 1 : System Architecture

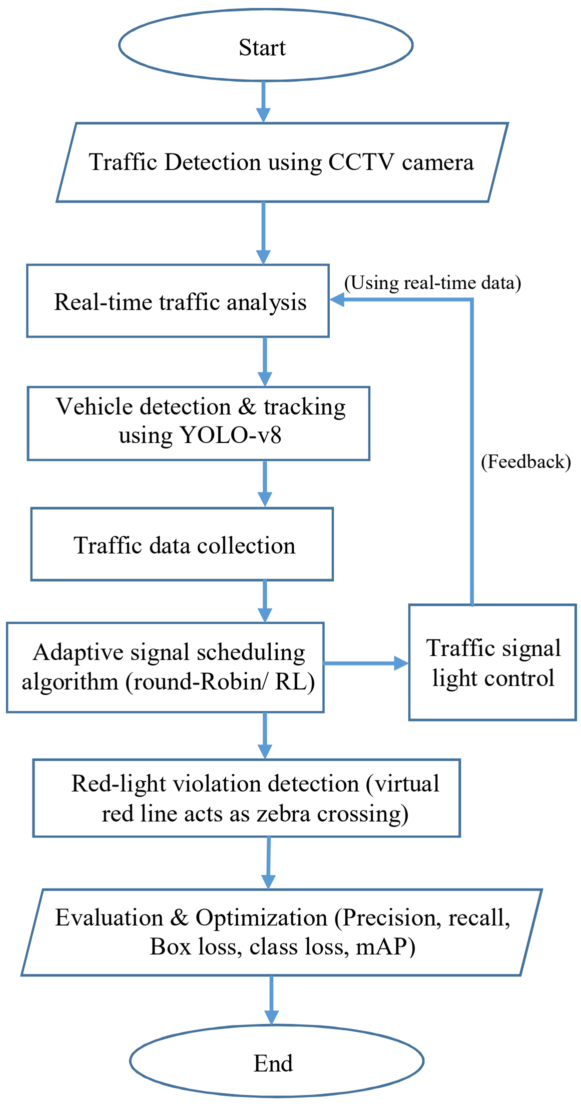

Start
Traffic Detection using CCTV camera
Real-time traffic analysis
(Using real-time data)
Vehicle detection & tracking using YOLO-v8
(Feedback)
Traffic data collection
Adaptive signal scheduling algorithm (round-Robin/ RL)
Traffic signal light control
Red-light violation detection (virtual red line acts as zebra crossing)
Evaluation & Optimization (Precision, recall, Box loss, class loss, mAP)
End

Fig 2 : System Flowchart

PyTorch facilitates the creation, training, and inference of models like DQN (Deep Q-Network) to make adaptive signal scheduling decisions. Its dynamic computational graph and strong GPU acceleration support make it ideal for experiments and real-time deployment.

NumPy is employed for efficient numerical computations, particularly for handling arrays, matrices, and mathematical operations involved in processing vehicle count data and managing the state and action spaces in reinforcement learning. It supports vectorized operations that enhance performance, especially when working with large-scale traffic datasets and image matrices.

Matplotlib is used for visualizing annotated traffic images, statistical charts, and performance metrics of the model. It helps developers understand the behaviour of the system, such as how signal timing affects traffic flow, and presents the results of the RL model's learning process. Visualization also aids in identifying issues and improving system design.

Reinforcement Learning (Q-Learning) is optionally integrated into the system to make the traffic signal scheduler intelligent and adaptive. Q learning learns from traffic data, identifying optimal signal timings that minimize congestion. Over time, the model improves its decisions through feedback, making the system robust and scalable for varying traffic scenarios.

Custom Dataset consists of traffic lane images, often simulated or captured from real-world CCTV footage. These datasets are used to train and test the YOLO model and simulate reinforcement learning environments. A well-labelled dataset ensures high accuracy in vehicle detection and provides realistic scenarios for optimizing signal scheduling strategies.

The **formulas used in YOLOv8** for key performance and loss metrics such as **precision**, **recall**, **box loss**, **class loss**, and **mAP (mean Average Precision)**.

1. Precision
Formula:

$$Precision = \frac{TP}{TP + FP} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(1)}$$

Notations:

- **TP** = True Positives (correct detections)
- **FP** = False Positives (wrong detections)

Meaning: Measures how many predicted bounding boxes are actually correct. High precision means few false positives.

2. Recall
Formula:

$$Recall = \frac{TP}{TP + FN} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2)}$$

Notations:

- **TP** = True Positives
- **FN** = False Negatives (missed detections)

Meaning: Measures how many actual objects were detected. High recall means few objects are missed.

3. Box Loss (Bounding Box Regression Loss)
Formula (YOLOv8 uses CIoU loss):

$$£_{\text{box}} = 1 - \text{CIoU} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3)}$$

CIoU (Complete Intersection over Union) is calculated as:

$$\text{CIoU} = \text{IoU} - \frac{\rho^2(b, b^{gt})}{c^2} - \propto v \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(4)}$$

Where:

- IoU = Intersection over Union between predicted and ground-truth boxes
- ρ = Euclidean distance between center points of predicted box b and ground-truth box b^{gt}
- c = diagonal length of the smallest enclosing box
- v = a measure of aspect ratio consistency
- α = a positive trade-off parameter

Meaning: CIoU considers overlap, center distance, and aspect ratio, making it more accurate than IoU loss alone.

4. Class Loss (Classification Loss)
Formula (Binary Cross Entropy):

$$\mathcal{L}_{cls} = \sum_{i=1}^{C} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \ldots\ldots\ldots \text{(5)}$$

Notations:

- C = Number of classes
- y_i = Ground truth label (0 or 1)
- p_i = Predicted probability for class i

Meaning: Penalizes wrong class predictions. Works well for multi-label classification, which is used in object detection.

5. mAP (mean Average Precision)
Formula:
$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i$ ……………………… *(6)*
Where **Average Precision (AP)** is:

$AP = \int_0^1 p(r)dr$ ……………………… *(7)*
Or approximated by:

$AP = \sum_n (R_n - R_{n-1}) P_n$ …………… *(8)*
Notations:

- N = Number of classes
- P_n, R_n = Precision and Recall at threshold n

Meaning: mAP is the average of APs across all classes. It summarizes the detector's overall performance at various IoU thresholds.

IV. RESULTS

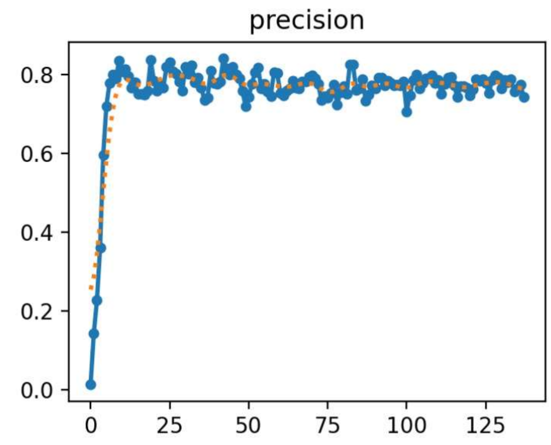


Fig 3 : Precision Graph for predicting bounding boxes.

This precision graph represents the training performance of the YOLOv8-based intelligent traffic signal and red-light violation detection system. Precision measures how accurately the model detects vehicles without false positives. In the early epochs, precision is low, but it rapidly increases within the first 10–15 epochs as the model learns to distinguish vehicles from the background. It then stabilizes around 0.75–0.8, indicating consistent and reliable performance. This high precision is crucial for the project, as it ensures accurate vehicle detection for dynamic signal scheduling and violation identification, enhancing the system's real-world effectiveness.

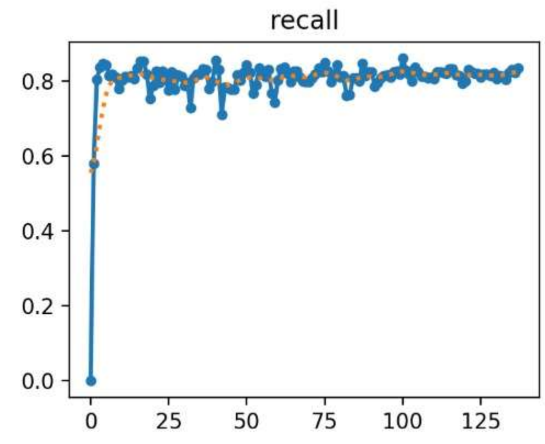


Fig 4 : Recall Graph for number of objects(vehicles) detected.

This recall graph illustrates the model's ability to correctly detect all relevant vehicles in the intelligent traffic signal scheduling and red-light violation detection system. Initially, the recall starts near zero, indicating the model is missing most vehicle detections. However, within just a few epochs, recall sharply increases to above 0.8, demonstrating rapid learning. Over the rest of the training period (up to 130+ epochs), recall remains consistently high with minor fluctuations, indicating the model reliably detects most vehicles present in the CCTV footage. High recall is essential for this project to ensure no vehicles are missed—particularly for accurate traffic density estimation and reliable red-light violation detection—ensuring real-time efficiency and enforcement accuracy.

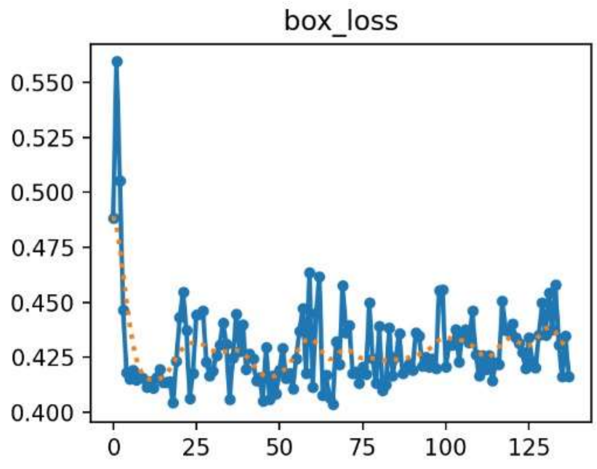


Fig 5 : Box loss (Bounding Box Regression Loss) Graph

This box_loss graph represents how accurately the YOLOv8 object detection model is predicting the location of vehicles within the frames of CCTV footage in the intelligent traffic signal and red-light violation detection system. Initially, the box loss is high (around 0.55), indicating that the bounding boxes predicted by the model are not closely matching the actual vehicle positions. However, within the first few epochs, the loss rapidly drops, showing the model is quickly learning to localize vehicles better. After that, the loss stabilizes and fluctuates slightly around 0.42, demonstrating consistent performance with minor variations. A lower and stable box loss signifies that the model can reliably place bounding boxes around vehicles, which is crucial for accurately counting traffic density and detecting red-light violations in real-time.

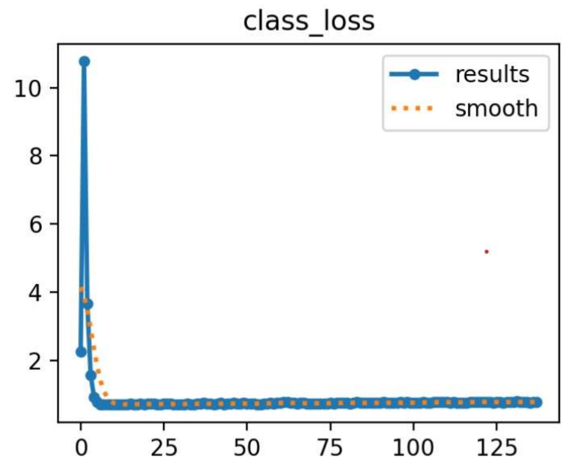


Fig 6 : Class Loss (Classification Loss) Graph

The graph depicting the class_loss, measures how accurately the model classifies different types of vehicles such as cars, bikes, buses, and trucks. Initially, the class loss is very high, indicating the model struggled to correctly classify vehicles during early training stages. However, it rapidly decreases and stabilizes at a low value after a few epochs, as reflected by both the raw and smoothed curves. This shows that the model effectively learned to classify vehicles accurately, which is crucial for the traffic density estimation and violation detection tasks in the system.

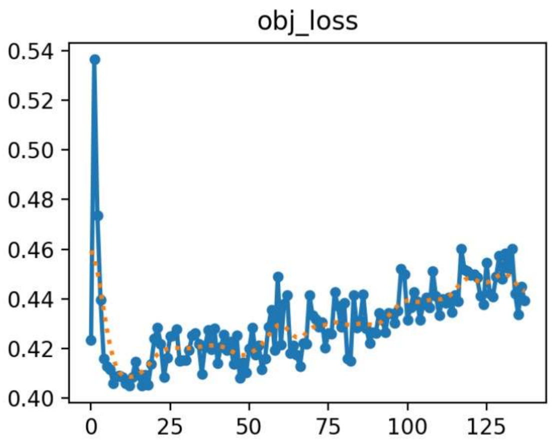


Fig 7 : Obj Loss (Object Loss) Graph

The graph shows the obj_loss, representing the model's confidence in predicting the presence of objects at specific locations. This loss also starts at a high value but quickly drops and stabilizes. Notably, there is a slight upward trend after around 40 to 50 epochs, along with some fluctuations. This behavior suggests that while the model initially learned to detect vehicle locations well, it began to show minor signs of overfitting as training progressed. Although this might slightly affect the precision of vehicle detection, the overall performance remains strong enough to support the traffic management objectives, including accurate violation monitoring. The graph shows the obj_loss, representing the model's confidence in predicting the presence of objects at specific locations. This loss also starts at a high value but quickly drops and stabilizes. Notably, there is a slight upward trend after around 40 to 50 epochs, along with some fluctuations. This behavior suggests that while the model initially learned to detect vehicle locations well, it began to show minor signs of overfitting as training progressed. Although this might slightly affect the precision of vehicle detection, the overall performance remains strong enough to support the traffic management objectives, including accurate violation monitoring.

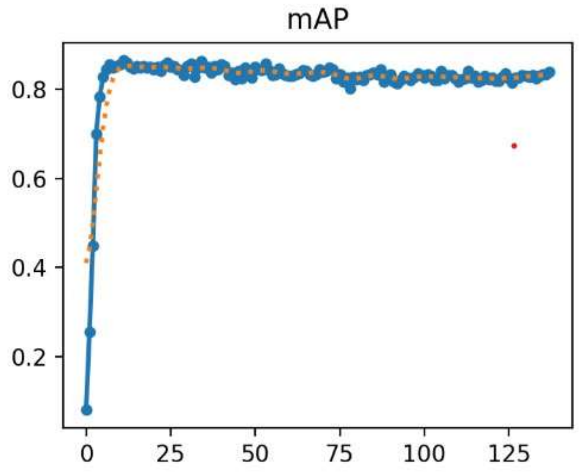


Fig 8 : Mean Average Precision Graph

The graph presents the mAP (mean Average Precision), which combines precision and recall to provide a comprehensive measure of the model's overall detection accuracy. The mAP value experiences a steep increase during the first few epochs, quickly surpassing 0.85, and then stabilizes at a high level throughout the rest of the training. This consistent high mAP score indicates that the model maintains strong object detection and classification performance across various vehicle types and traffic scenarios. Such high accuracy is vital for the adaptive signal control and law enforcement components of the system, ensuring that traffic congestion is minimized and red-light violations are reliably detected.

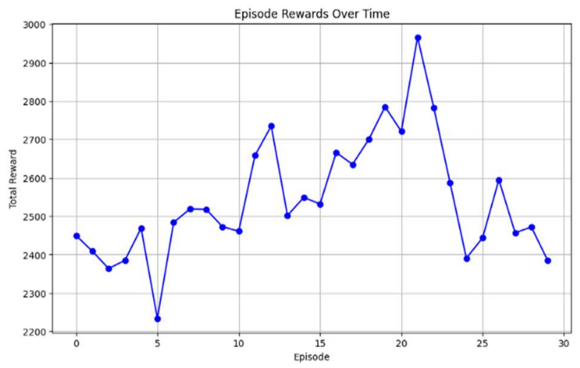


Fig 9 : Reinforcement Learning (RL) Graph

This graph depicts the total episodic rewards over the course of 30 episodes in a reinforcement learning (RL) training process. The x-axis represents the episode number, while the y-axis shows the cumulative reward obtained in each episode. The curve, which is plotted with blue markers connected by lines, captures the agent's performance progression as it interacts with the environment. Initially, the rewards fluctuate within a lower range (approximately between 2350 and 2500), reflecting the exploratory nature of early episodes where the policy is still being optimized. As training advances, there is a noticeable upward trend, with several peaks indicating episodes where the agent successfully exploits its learned policy to achieve higher rewards — for instance, around episode 12 and a significant peak near episode 21 (approaching a total reward of ~2970). However, the variability and occasional dips (e.g., episode 5 and post-episode 21) indicate that the policy is still subject to instability or suboptimal exploration-exploitation trade-offs. The oscillations toward the end suggest either a non-stationary environment, high variance in policy actions, or insufficient convergence of the learning algorithm. Overall, the graph implies gradual policy improvement with episodic variability, which is typical in many RL algorithms prior to achieving stable convergence.

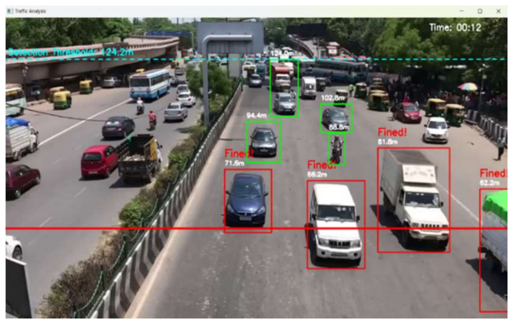


Fig 10 : Vehicle detection and tracking

This image demonstrates a vehicle detection and tracking system implemented using the YOLOv8 (You Only Look Once version 8) object detection algorithm, applied in the context of traffic monitoring and automated violation enforcement. The scene shows a busy multilane road segmented for analysis. Detected vehicles are bounded by colored boxes: green boxes indicate compliant vehicles, while red boxes denote vehicles flagged for a violation, specifically for breaching a predefined detection threshold zone. The cyan dashed line near the horizon, labeled "Detection Threshold: 124.2m", serves as a virtual geofencing line — vehicles crossing this line and detected closer to the camera (with a lower distance value) are subject to scrutiny for potential violations, such as restricted lane usage or signal jumping. Each bounding box is annotated with a distance value (in meters), which is likely computed using monocular depth estimation or homography transformation techniques to estimate how far the vehicle is from the camera. Vehicles within a certain proximity (under the detection threshold) and flagged as violators are marked explicitly with the label "Fined!" alongside their respective distances (e.g., 71.6m, 66.2m). The temporal component of the video analysis is indicated by the "Time: 00:12" timestamp on the top right, suggesting that the system performs real-time detection and tracking. YOLOv8's fast inference speed and improved object localization capabilities enable robust detection across various vehicle types (cars, motorcycles, trucks, etc.), supporting intelligent traffic analysis and automated law enforcement applications efficiently.

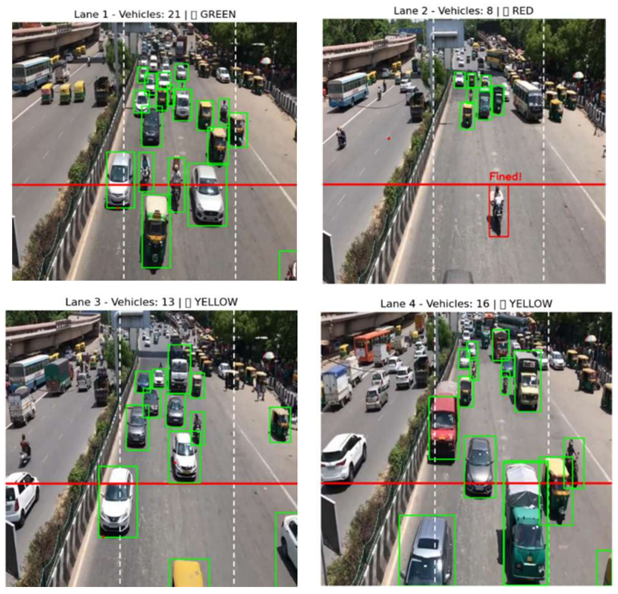
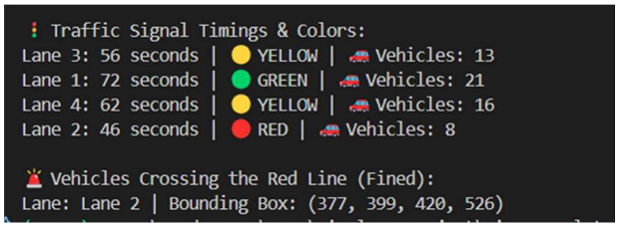

Lane 1 - Vehicles: 21 | ▯ GREEN
Lane 2 - Vehicles: 8 | ▯ RED
Fined!
Lane 3 - Vehicles: 13 | ▯ YELLOW
Lane 4 - Vehicles: 16 | ▯ YELLOW

Fig 11 : Vehicle density count

In Lane 1, the system has detected the highest number of vehicles (21), resulting in a green signal, allowing traffic to flow. This indicates that the system prioritizes lanes with heavier congestion. Lane 2 shows a lighter traffic load with only 8 vehicles, and the signal is red. However, a motorcycle has crossed the designated red stop line, triggering a "Fined!" warning, which showcases the system's ability to detect and enforce red light violations using computer vision. Lane 3 contains 13 vehicles and is under a yellow signal, suggesting that it is either preparing to stop or is next in queue for a green light. Similarly, Lane 4 has 16 vehicles and is also under a yellow signal, indicating that it may soon receive a green light once Lane 1 is cleared. The system uses object detection to count vehicles in real-time, marks them with green bounding boxes, and enforces rules by identifying violators using red lines and alerts. This intelligent setup demonstrates how adaptive signal control and automated violation detection can effectively manage traffic flow and improve road safety at busy intersections.



```
 🚦 Traffic Signal Timings & Colors:
Lane 3: 56 seconds | 🟡 YELLOW | 🚗 Vehicles: 13
Lane 1: 72 seconds | 🟢 GREEN | 🚗 Vehicles: 21
Lane 4: 62 seconds | 🟡 YELLOW | 🚗 Vehicles: 16
Lane 2: 46 seconds | 🔴 RED | 🚗 Vehicles: 8

 🚨 Vehicles Crossing the Red Line (Fined):
Lane: Lane 2 | Bounding Box: (377, 399, 420, 526)
```

Fig 12 : Signal timers as per density count

The image displays the output of a smart traffic management system that dynamically adjusts traffic signal timings based on the number of vehicles in each lane. It shows the signal duration, current signal color, and vehicle count for four lanes. Lane 1, having the highest number of vehicles (21), is given a green signal for 72 seconds to allow smooth traffic flow. Lane 3 and Lane 4, with 13 and 16 vehicles respectively, are currently on yellow signals with durations of 56 and 62 seconds. Lane 2, which has the least number of vehicles (8), is on a red signal for 46 seconds. Additionally, the system detects red light violations. It has identified a vehicle in Lane 2 that crossed the red signal line, and the vehicle's position is highlighted using a bounding box with coordinates (377, 399, 420, 526). This indicates the system's ability to not only manage signal timings based on traffic density but also enforce traffic rules by detecting violations in real time.

Metric	**Initial Value**	**Final Value**	**Remarks**
Class Loss	1050%	100%	Sharp drop in early epochs; stabilizes consistently.
Box Loss	56%	42%	Shows fluctuation but gradually declines and stabilizes.
Object Loss	54%	44%	Initially drops, later fluctuates slightly upward.
Precision	10%	78%	Rapid increase in early epochs; remains high and stable.
Recall	0%	82%	Steep rise followed by stabilization.
mAP@0.5	0%	85%	Fast initial growth; plateaus around epoch 20.
mAP@0.5:0.95	0%	54%	Slower to converge but remains steady after early training.

Fig 13 : Model Training Performance Metrics

The model training performance metrics provide a comprehensive view of how the object detection model evolved over time. Class loss started very high at 1050%, indicating poor initial classification accuracy, but dropped sharply to 100%, showing rapid improvement in identifying object classes correctly. Similarly, box loss, which measures the error in predicting object bounding box coordinates, decreased gradually from 56% to 42%, reflecting the model's growing precision in object localization. Object loss, which quantifies how well the model detects the presence of objects, also reduced from 54% to 44%, suggesting consistent improvement in recognizing object instances. The precision rose significantly from 10% to 78%, meaning the model increasingly made correct positive predictions. Recall, starting at 0%, improved to 82%, indicating the model's growing ability to detect most of the actual objects present in the images. The mean Average Precision at IoU 0.5 (mAP@0.5) increased from 0% to 85%, highlighting a high level of accuracy in object detection. Meanwhile, mAP@0.5:0.95, a stricter metric that averages performance over multiple IoU thresholds, improved to 54%, demonstrating the model's robustness and balanced performance across various levels of localization accuracy. Overall, these metrics collectively showcase the successful training progression and the model's effectiveness in object detection tasks.

V. CONCLUSIONS

Managing traffic at busy intersections in growing cities has become one of the city's biggest challenges. Traditional signal systems operating with fixed timers are not designed in a way that bypasses in real traffic conditions. As a result, they often cause long waits, waste of fuel, and frustration among commuters. To improve this, the proposed system uses an intelligent, AI-controlled approach that brings automation and intelligence to traffic control. A deep learning model known for its speed and accuracy, Yolov8 is used to detect vehicles in real time. It helps you identify different types of vehicles, cars, bicycles, buses and trucks from CCTV film material that live at intersections. OpenCV is used for image processing tasks. Python acts as the basic language for scripting and integration for all components, while Pytorch is used specifically to implement reinforcement learning (RL) with DQN algorithms (Deep Q-Network). More vehicle traces have more time, and lanes with less traffic will be removed soon. This adaptive control reduces unnecessary delays, improves vehicle flow and saves fuel. A red light violation is automatically recognized and the system can record and identify vehicle number plates without human participation. This helps in quick and accurate enforcement. Overall, the integration of technologies such as Yolov8, opencv, reinforcement learning, and practical analysis of this system demonstrates how to effectively use artificial intelligence to solve everyday urban problems. We provide scalable, inexpensive and intelligent solutions for intelligent cities aimed at building better, faster and safer transportation systems.

REFERENCES

[1] Kranti Shingate, Komal Jagdale, Yohann Dias, ”Adaptive Traffic Control System Using Reinforcement Learning,” International Journal of Engineering Research & Technology, Vol. 9 Issue 02, February-2020

[2] Vineeta Choudhary, Shery Tank, Kriti Gulati, Shubham Jani, Bhaumik Machhi ” Adaptive Traffic Control System”, International Journal for Research in Applied Science & Engineering Technology, Volume 11 Issue II Feb 2023

[3] Jiri Cejkaa, Josef Šedivýa, ”Discussion of Operational Transport Analysis Methods and the Practical Application of Queuing Theory to Stationary Traffic”, Transportation Research Procedia 53 (2021)

[4] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, Jianzhong, ” GMAN: A Graph Multi-Attention Network for Traffic Prediction”, The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20).

[5] B. Liu, C. -T. Lam and B. K. Ng, "Improved Real-Time Traffic Congestion Detection with Automatic Image Cropping using Online Camera Images," 2021 IEEE 21st International Conference on Communication Technology (ICCT), Tianjin, China, 2021, pp. 1117-1122, doi: 10.1109/ICCT52962.2021.9657853.

[6] Bowie Liu, Chan-Tong Lam, Benjamin k. Ng, Xiaochen Yuan, Sio Kei Im “A Graph-Based Framework for Traffic Forecasting and Congestion Detection Using Online Images From Multiple Cameras” 1 January 2024, Doi: 10.1109/ACCESS.2023.3349034

[7] M. G. R. Alam et al., "Queueing Theory Based Vehicular Traffic Management System Through Jackson Network Model and Optimization," in IEEE Access, vol. 9, pp. 136018-136031, 2021, doi: 10.1109/ACCESS.2021.3116503.

[8] Moolchand Sharma1, Ananya Bansal, Vaibhav Kashyap, Pramod Goyal, Dr.Tariq Hussain Sheikh “Intelligent Traffic Light Control System Based On Traffic Environment Using Deep Learning” doi:10.1088/1757-899X/1022/1/012122.

[9] Mortazavi Azad, Seyedeh M., and A. Ramazani. "Smart control of traffic lights based on traffic density in the multi-intersection network by using Q learning." *Discover Artificial Intelligence* 3, no. 1 ,6 November 2023..

[10] Ravina Dnyaneshwar Chavhan, Dr. G.B. Sambare, ” AI-Driven Traffic Management Systems In Smart Cities: A Review,” Educational Administration: Theory and Practice 2024, 30(5), 105-116.

[11] Abhijeet Choudhary, Akash Gupta, Akshay Dhuri, Prof. Nilima Nikam, ”Artificial Intelligence Based Smart Traffic Management System Using Video Processing, International Research Journal of Engineering and Technology Volume: 05 Issue: 03 | Mar-2018

[12] Khalid Mohammed Almatar ” Implementing AI-Driven Traffic Signal Systems for Enhanced Traffic Management in Dammam International Journal of Sustainable Development and Planning Vol. 19, No. 2, February, 2024, pp. 781-790

[13] Rajeshwari Sundar, Santhoshs Hebbar, and Varaprasad Golla “Implementing Intelligent Traffic Control System” Sensor Journal, IEEE(Volume:15, Issue :2)

[14] Md. Rokebul Islam*, Nafis Ibn Shahid*, Dewan Tanzim ul Karim*, Abdullah Al Mamun**, Dr. Md. Khalilur Rh “An Efficient Algorithm for Detecting Traffic Congestion and a Framework for Smart Traffic Control System” Journal, IEEE Jan 20.

[15] Z. Liu, W. Han, H. Xu, K. Gong, Q. Zeng, and X. Zhao, "Research on vehicle detection based on improved YOLOX_S," Scientific Reports, vol. 13, no. 1, p. 23081, 2023.