International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Smart Signals, Smoother Streets: Dynamic Traffic Signal Control

Author Details:

Mr. R.S. Derle[1], Mr. P.R. Pachorkar[2], Mast. Aditya Jadhav[3], Mast. Aditya Nikam[4], Mast. Pushkaraj Jadhav[5], Mast. Abhishek Baste[6]

[1] Information Technology Department, MVPS's KBTCOE, Nashik, India

[2] Information Technology Department, MVPS's KBTCOE, Nashik, India

[3] Information Technology Department, MVPS's KBTCOE, Nashik, India

[4] Information Technology Department, MVPS's KBTCOE, Nashik, India

[5] Information Technology Department, MVPS's KBTCOE, Nashik, India

[6] Information Technology Department, MVPS's KBTCOE, Nashik, India

Corresponding Author Email: nikamaditya333@gmail.com | Information Technology Department, MVPS's KBTCOE, Nashik, India

https://doi.org/10.55041/ijsmt.v2i4.248

Cite this Article: Derle, R., Pachorkar, P., Jadhav, M. A., Nikam, M. A., Jadhav, M. P. & Baste, M. A. (2026). Smart Signals, Smoother Streets: Dynamic Traffic Signal Control. International Journal of Science, Strategic Management and Technology, *02*(04). https://doi.org/10.55041/ijsmt.v2i4.248

Abstract—

Traffic congestion is a major problem in modern urban areas due to the increasing number of vehicles and the limitations of traditional fixed-time traffic signal systems. These conventional systems fail to respond to real-time traffic conditions, resulting in longer waiting times, higher fuel consumption, and increased environmental pollution. This project presents a Dynamic Traffic Signal Allocation System that combines computer vision and real-time data processing to optimize traffic signal control. The system integrates the SUMO traffic simulator to model road networks and simulate traffic flow, while a YOLO-based object detection model is used to detect and count vehicles from simulated video streams. Based on the detected vehicle counts, a Q-Learning algorithm determines the optimal green signal duration for each lane by considering current traffic density and minimizing overall vehicle delay. A TraCI interface is used to communicate with the SUMO simulation in real time and control signal phase operations. Traffic data and system outputs are stored using Firebase Firestore, enabling realtime monitoring across sessions. A Flask-based backend handles data processing and serves the application, while a web-based dashboard built with HTML, CSS, and JavaScript displays traffic statistics through charts and exportable reports. The proposed system demonstrates improved traffic flow, reduced congestion at intersections, and a practical approach to adaptive signal control. It is designed with scalability in mind and shows potential for integration into broader smart city frameworks as a cost-effective alternative to conventional traffic

management reflect the core contribution of the paper. Please avoid citations in the abstract.

Keywords— Adaptive traffic signal control, reinforcement learning, Q-learning, computer vision, YOLO, traffic flow optimization, SUMO simulation, intelligent transportation systems, real-time traffic monitoring, cloud-based traffic management, smart traffic systems.

I. INTRODUCTION

Traffic congestion has become a serious issue in urban areas due to the rapid increase in the number of vehicles. Traditional traffic signal systems operate on fixed time intervals and do not consider real-time traffic conditions. As a result, some roads remain overcrowded while others have very less traffic, leading to increased waiting time, fuel wastage, and environmental pollution. To solve this problem, this project proposes a Dynamic Traffic Signal Allocation System that uses Artificial Intelligence (AI), Computer Vision, and real-time data processing. The main goal of the system is to automatically adjust traffic signal timing based on the actual traffic density at an intersection. In this project, vehicle detection is performed using a YOLO (You Only Look Once) deep learning model. The model was trained and tested using a cloud-based environment where different traffic datasets were used to improve detection accuracy. During training, the model learns to identify various types of vehicles such as cars, bikes, and trucks. After training, the model generates weights that are used for real-time detection in the system. The overall working of the system involves multiple integrated components. First, a traffic environment is created using a simulation tool that models road networks and vehicle movement. This simulation generates a video stream representing real-time traffic conditions. The video stream is then processed by the trained YOLO model, which detects and counts vehicles in each lane. The detected traffic data is passed to a Q-Learning based algorithm, which acts as the decision-making component of the system. This algorithm analyzes the traffic conditions and selects the optimal signal timing to reduce congestion and waiting time. The system continuously learns and improves its decisions based on traffic patterns. A communication interface is used to connect the simulation with the decision-making system, allowing real-time control of traffic signals. Additionally, all traffic data is stored and synchronized using a real-time database, which enables monitoring and analysis. A backend server processes the data, and a web-based dashboard is used to visualize traffic conditions, analytics, and reports. By combining simulation, computer vision, and machine learning techniques, the proposed system provides an intelligent and adaptive approach to traffic management. It improves traffic efficiency, reduces delays, and has strong potential for implementation in smart city infrastructure.

II. LITERATURE REVIEW

This section presents the key AI-based approaches studied during the literature review phase of this project.

A. Computer Vision:

Computer Vision enables machines to interpret visual data such as images and videos. In traffic systems, object detection models like YOLO process video frames to detect and count vehicles in real time by drawing bounding boxes around identified objects. The per-lane vehicle count is then extracted as structured data for further processing. This approach removes the dependency on physical road sensors, is cost-effective, and can be deployed using only a standard camera feed in both simulated and real-world environments.

B. Multi-Agent Reinforcement Learning (MARL):

Multi-Agent Reinforcement Learning involves multiple agents, each representing a traffic signal at an intersection. Each agent observes its local traffic state and selects a signal action independently, while coordination mechanisms allow neighboring agents to share information for network-level optimization. Methods such as Co Light and QMIX have demonstrated reductions in average travel time on real city datasets by enabling cooperative signal control across multiple intersections simultaneously

C. Deep Q-Network (DQN) Deep Q-Network:

replaces the traditional Q-table with a neural network to approximate Q-values, allowing the algorithm to handle large and complex state spaces. In traffic control, the state is defined by lane-wise vehicle counts, the action is the selection of a signal phase, and the reward measures delay reduction. The network is trained using experience replay and a target network to stabilise learning, and continuously improves signal timing decisions over repeated simulation cycles

D. Graph Coordinated Deep Reinforcement Learning:

This approach represents the road network as a graph where intersections are nodes and road segments are edges. A Graph Attention Network (GAT) allows each intersection agent to attend selectively to information from its neighbors based on current traffic conditions. This produces a context-aware state for each agent that captures both local and network-level patterns, making it effective for city-scale traffic management on irregular road topologies.

E. WHY WE CHOSE COMPUTER VISION AND Q-LEARNING:

Among the approaches studied, Computer Vision and Q Learning were selected based on their practical suitability and alignment with the available simulation environment. Computer Vision using YOLO provides real-time vehicle detection directly from video frames without requiring physical road sensors. It integrates naturally with SUMO, which outputs a video stream, making the transition from detection to per-lane counts straightforward. Q-Learning was chosen because it suits the discrete action space of traffic signal control, where the system selects from a defined set of green durations. It operates using a Q-table that is practical to implement and interpret within a simulation, and improves timing decisions based on observed reductions in waiting time over repeated cycles. Together, YOLO observes the traffic state, Q-Learning acts on it, and TraCI applies the decision back to the simulation – forming a complete, modular, and testable detection-to-decision pipeline.

III. METHODOLOGY

A. Overall Architecture

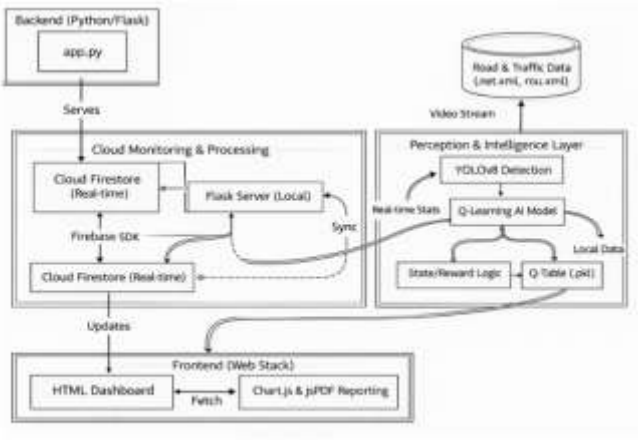

Fig. 1. Architecture Diagram

The proposed Dynamic Traffic Signal Allocation System is built on a modular architecture that connects simulation, vehicle detection, signal control, data storage, and visualization into a single working pipeline. The system is divided into two main parts: the backend and the frontend. The backend handles all computational tasks including vehicle counting, traffic data processing, and signal timing decisions, while the frontend provides visualization, simulation display, and user interaction through a web-based dashboard.

The overall workflow begins with SUMO generating a simulated traffic environment that produces a video stream of vehicle movement across lanes. This stream is processed by the trained YOLO model, which detects and counts vehicles in each lane in real time. The counts are passed to a Q-Learning based controller, which evaluates current lane densities and assigns green signal durations accordingly. The TraCI interface relays these timing decisions back to the SUMO simulation, completing the control loop.

All traffic records and signal timing outputs are stored in Firebase Firestore, allowing the data to be retrieved and displayed on the dashboard without delay. The Flask backend coordinates communication between the detection module, the signal controller, and the database. The frontend dashboard, built with HTML, CSS, and JavaScript,

presents lane-wise counts, signal states, and session summaries through charts and tables.

In addition, a hardware demonstration using a Raspberry Pi is included to physically represent signal switching, showing how the software decisions can be mapped to real-world signal outputs. The modular design of the system ensures that each component can be updated or replaced independently, and the architecture can be extended to handle multiple intersections as needed.

B. Backend Implementation

The backend of the proposed system acts as the core processing unit where all major operations such as traffic data analysis, vehicle detection, and signal timing decisions are performed. It is implemented using Python and integrates multiple components including the SUMO simulation interface, trained machine learning models, and Firebase Firestore for cloud storage. The backend continuously interacts with the traffic simulation to collect real-time data, processes this data using trained models, and generates optimized traffic signal decisions. It also manages communication with the frontend dashboard and cloud database to ensure real-time monitoring and synchronization. By combining data processing with model-based decision-making, the backend enables dynamic and adaptive traffic control across the system.

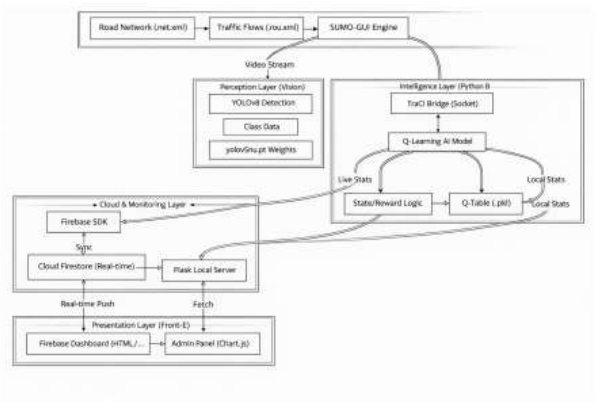

Fig. 2. Backend Diagram

1) Detection Model (Working and Training):

The detection model is an essential component of the backend system, responsible for identifying and counting vehicles from traffic images or video streams. In this project, a real-time object detection approach based on the YOLOv5 architecture is used. YOLO, which stands for "You Only Look Once," is a single stage object detection algorithm that processes the entire image in a single pass and predicts bounding boxes along with class probabilities. Due to its high speed and accuracy, YOLO is highly suitable for real-time traffic monitoring applications. The working of the detection model begins with input processing, where traffic frames obtained from the simulation environment or camera feed are resized to a standard resolution of 640 × 640 pixels. This ensures uniformity and improves model performance. These input images are then passed through the backbone of the YOLO model, which extracts important spatial features such as edges, shapes, and patterns associated with different types of vehicles. The model then performs object detection by predicting bounding boxes around vehicles, identifying their class labels, and assigning confidence scores to each detection. The system is designed to detect multiple vehicle categories commonly found in Indian traffic conditions, including cars, three-wheelers, buses, trucks, motorbikes, and vans. After detection, the model counts the number of vehicles present in each lane and generates traffic density information. This output is then forwarded to the decision-making model, where it is used as input for optimizing traffic signal control. This detection pipeline enables accurate and real-time estimation of traffic conditions, which is critical for implementing adaptive signal systems. To improve the performance of the detection model, it was fine-tuned using a custom dataset specifically designed for traffic scenarios. The dataset used for training is the DATS3K Traffic Dataset, which contains a total of 3000 images. Out of these, 2100 images were used for training and 900 images were used for validation. The dataset includes diverse traffic scenarios with mixed vehicle types, making it suitable for real-world applications. All annotations were provided in YOLO format, allowing seamless integration with the training framework. The model configuration is

based on the YOLOv5 Nano variant, which is lightweight and efficient for real-time applications. The training was performed using the Ultralytics framework with an input image size of 640×640 pixels and a batch size of 16. The model was trained for 80 epochs using an automatically selected AdamW optimizer. The training process was accelerated using a CUDA-enabled NVIDIA Tesla T4 GPU, which significantly reduced computation time.

During training, pretrained weights were used as the initial model parameters to improve convergence and reduce training time. The dataset was loaded and cached in memory to enhance training efficiency. Various data augmentation techniques were applied, including horizontal flipping, color adjustments, and image enhancement methods, to improve the model's generalization capability. The training process involved optimizing multiple loss functions, including box loss for localization accuracy, classification loss for correct labeling, and distribution focal loss for better bounding box prediction.

2) Decision Model (Working and Training using DQN):

The decision-making component of the system is based on a Deep Q-Network (DQN), which is an advanced reinforcement learning algorithm used to optimize traffic signal control. The objective of this model is to dynamically decide which traffic signal should be given priority based on current traffic conditions, thereby minimizing congestion and waiting time. The model is trained using a reinforcement learning approach where the system interacts with the simulated environment and learns from experience. During training, the agent observes the current traffic state, selects an action, and receives a reward based on the effectiveness of that action. Over time, the model learns an optimal policy that maximizes the cumulative reward. In this implementation, the training process was carried out for 800 episodes, with each episode consisting of a maximum of 200 steps. The training was performed on a GPU enabled environment to improve computational efficiency. During training, the exploration-exploitation strategy is controlled using an epsilon-greedy approach, where the epsilon value gradually reduces to encourage the model to exploit learned knowledge.

The working of the decision model is based on the concept of states, actions, and rewards. The state of the system is defined by the traffic density at each direction of the intersection, along with the current active signal phase. To simplify the state representation, the vehicle counts are discretized into different levels such as low, medium, and high traffic. This helps in reducing the complexity of the state space and improves learning efficiency. The action space consists of two possible actions: either to continue the current green signal or to switch to the next signal in a predefined sequence. The model selects actions using an epsilon-greedy strategy, where it either chooses a random action for exploration or selects the best action based on learned Q-values. The reward function is designed to encourage efficient traffic flow. When the system reduces the number of waiting vehicles in the current lane, a positive reward is given. However, if congestion increases in other lanes or unnecessary switching occurs, a negative reward is assigned. This ensures that the model learns to balance traffic across all directions.

The learning process updates the Q-values using the Bellman equation, where the current Q-value is updated based on the reward received and the expected future rewards:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (1)$$

where s is the current state, a is the selected action, r is the reward received, α is the learning rate, γ is the discount factor, and s ′ is the resulting next state. This allows the model to improve its decision-making over time. The Q-table is stored and updated continuously, ensuring that learned knowledge is preserved even after the simulation ends. In real-time operation, the decision model receives vehicle count data from the detection system through the simulation interface. It evaluates the current traffic condition and determines whether to maintain the

current signal or switch to another direction. The system also incorporates additional logic such as minimum green time, maximum green time, and priority handling to ensure realistic and safe traffic signal transitions. The transition between signals includes yellow and all-red phases to maintain safety at the intersection. Furthermore, the system supports multi-junction coordination, where each traffic signal operates as an independent agent but follows a similar learning strategy. The decisions made by the model are continuously sent to the simulation environment and also updated on the dashboard through the backend server.

3) Data Storage:

Data storage plays an important role in the proposed system by enabling real-time data synchronization, monitoring, and analysis. In this project, Firebase Fire store is used as the cloud-based database to store and manage all traffic-related information generated by the backend during simulation. The backend continuously collects traffic data from the simulation environment, including vehicle counts, waiting vehicles, signal phases, and traffic conditions at each junction. This data is processed and transmitted to Fire store in real time. Each traffic junction is treated as a separate entity in the database, and its corresponding data is stored in a structured format that includes directional vehicle counts for north, east, south, and west directions, along with waiting times, current signal states, and the total number of vehicles present in the system at each update cycle. The cloud database ensures that all updates are synchronized instantly, allowing the frontend dashboard to fetch and display the latest traffic information without delay. This real-time data flow is essential for monitoring traffic conditions and evaluating system performance during and after simulation runs. Additionally, storing data in the cloud provides scalability, reliability, and easy access from different components of the system without requiring a local database setup. The integration of Firebase also supports timestamp-based updates, which help in tracking traffic patterns over time. This stored data can further be used for analysis, performance evaluation, and future improvements in the traffic management system. Overall, the data storage module acts as a bridge between the backend processing and frontend visualization, ensuring seamless communication and efficient data handling across all components

C. Demonstration:

Raspberry Pi and LED Implementation To demonstrate the practical applicability of the proposed system, a hardware-based prototype is implemented using a Raspberry Pi and LED lights. This setup provides a physical representation of how the dynamic traffic signal system can be deployed at an actual intersection, bridging the gap between the software simulation and real-world signal control. The Raspberry Pi acts as a microcontroller unit that receives traffic signal decisions from the backend system. It is connected to LED lights through its GPIO (General Purpose Input Output) pins, where each LED represents a specific traffic signal state, namely red, yellow, and green. Based on the output generated by the decision model, the Raspberry Pi updates the LED signals accordingly, replicating the behavior of a real traffic light at an intersection. The communication between the backend and the Raspberry Pi is established through network-based data transfer, where the processed signal decisions are transmitted to the hardware unit in real time. The Raspberry Pi interprets the received signal state data and drives the corresponding GPIO pins to switch the LEDs on or off. This allows the hardware setup to physically demonstrate how traffic signals change dynamically in response to detected traffic conditions, in sync with the decisions made by the DQN-based controller. The hardware implementation enhances the overall project by demonstrating that the proposed system is not limited to a virtual simulation environment but can be extended to actual traffic infrastructure. The use of Raspberry Pi makes the prototype cost-effective, compact, and easy to replicate, making it suitable for scalable deployment across multiple intersections. The LED indicators provide a clear and immediate visual representation of signal transitions, including green, yellow, and red phases, which mirrors the operation of real traffic lights.

D. Frontend Implementation

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

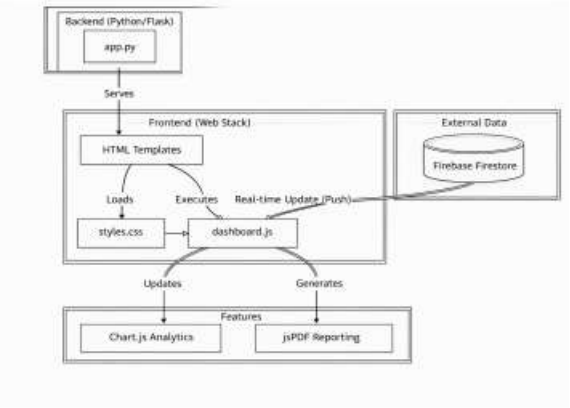

Fig. 3. Frontenf Diagram

The frontend of the proposed system is designed to provide an interactive and visual representation of the traffic management process. It acts as a bridge between the backend processing and the end user by presenting complex traffic data in an understandable and user-friendly manner. The frontend not only displays real-time traffic conditions but also helps in monitoring system performance, analyzing traffic patterns, and validating the effectiveness of the implemented algorithms. It integrates simulation and dashboard visualization components to provide a complete overview of the system's operation. The frontend is structured in such a way that it continuously receives processed data from the backend through cloud integration and updates the user interface dynamically. This ensures that any changes in traffic conditions or signal decisions are immediately reflected on the screen. The combination of simulation display and dashboard visualization enhances the usability of the system and allows users to observe how the signal timing decisions affect overall traffic flow in real time. 1) SUMO Simulation (Two Junction System): The SUMO (Simulation of Urban Mobility) platform is used as the core simulation environment to model and analyze traffic behavior. It is an open-source microscopic traffic simulator that allows detailed modeling of road networks, vehicle movement, and traffic signal control. In this project, a two-junction traffic network is implemented to simulate real-world urban traffic scenarios where multiple intersections are interconnected.

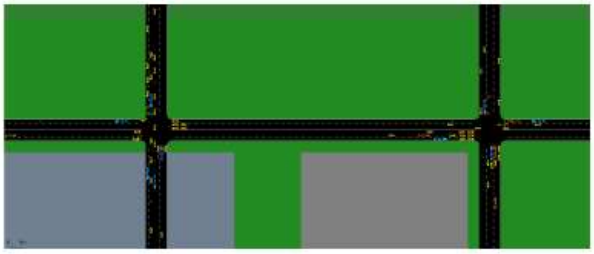

Fig. 4. SUMO Simulation.

Each junction in the simulation is designed with multiple incoming and outgoing lanes, representing different directions of traffic flow such as north, east, south, and west. The simulation uses predefined network and route configuration files to define road topology and vehicle paths. Vehicles of various types are generated dynamically within the system, creating different traffic densities and congestion patterns. This variability is important for testing the adaptability of the system under changing traffic conditions. The two junctions are interconnected to demonstrate coordinated traffic management. Instead of treating each intersection independently, the system considers the interaction between junctions, allowing better distribution of traffic flow across the network. Traffic data such as vehicle count, queue length, and waiting time is continuously extracted from the simulation using the TraCI interface, which enables real-time communication between the simulation and the backend processing modules.

Based on the decisions generated by the backend decision model, traffic signals in the simulation are updated dynamically. The simulation also incorporates realistic signal transitions, including green, yellow, and all-red phases, to ensure safe and practical traffic control. The use of SUMO provides a controlled environment where different traffic scenarios can be tested repeatedly, making it highly effective for training, evaluation, and performance analysis of the system. 2) Dashboard: The dashboard is an essential component of the frontend that provides real-time visualization and analysis of traffic data. It is developed using web technologies including HTML, CSS, and JavaScript, along with specialized libraries for data visualization and reporting. The dashboard serves as a monitoring interface where users can observe system behavior,

analyze traffic patterns, and evaluate the effectiveness of the traffic signal control strategy at each junction.

Fig. 5. Dashboard

The dashboard retrieves data from Firebase Fire store, where it is continuously updated by the backend during simulation runs. This ensures that the displayed information always reflects the current state of the traffic system. Key parameters such as vehicle counts, waiting vehicles, signal phases, and total traffic load are presented in both numerical and graphical formats. Visualization techniques including charts and graphs are used to simplify complex traffic data and make it easier to interpret for monitoring purposes. In addition to real-time monitoring, the dashboard provides analytical capabilities that help in understanding traffic trends over the course of a simulation session. It allows users to track system performance over time and evaluate how effectively congestion is being managed across both junctions. The integration of reporting tools enables the generation of summaries and session reports, which can be used for further analysis or documentation purposes. The dashboard is designed to be responsive and interactive, ensuring a smooth user experience across different devices and screen sizes. It updates automatically without requiring manual page refresh, providing a seamless real-time monitoring experience. By combining data visualization with continuous cloud-based updates, the dashboard enhances the overall functionality of the system and plays a crucial role in demonstrating the practical benefits of the proposed adaptive traffic management approach.

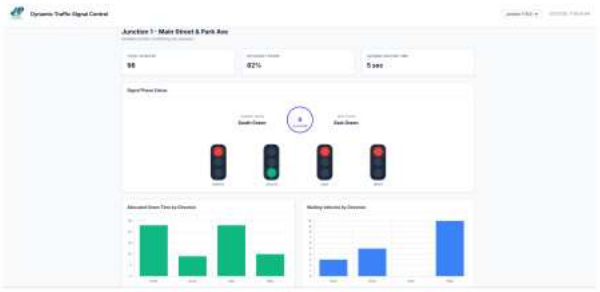

Fig. 6. Indivisual Junction Dashboard.

IV. RESULTS AND DISCUSSION

The performance of the model was evaluated after each training epoch using the validation dataset. The final trained model achieved high accuracy, with precision of 0.974, recall of 0.945, mAP@50 of 0.981, and mAP@50–95 of 0.913. These results indicate that the model is highly effective in detecting vehicles across different categories. Slight variations in performance were observed for smaller objects such as motorbikes due to their size and variability in appearance.

In terms of inference performance, the model is capable of processing images in approximately 1.8 milliseconds per frame, making it suitable for real-time deployment. The optimized YOLOv5 Nano model provides a balance between speed and accuracy, ensuring efficient operation in dynamic traffic environments.

The training results indicate that the Decision-making model gradually stabilizes as learning progresses. Initially, the reward values are low due to random exploration, but as the model learns, the average reward improves and becomes more consistent. The loss values also decrease over time, indicating that the model is successfully minimizing prediction errors. The trained model is periodically saved during training, and the final model is stored as a file for deployment.

The proposed system can be extended for real-world deployment by integrating CCTV cameras and edge computing devices such as Jetson Nano to enable live traffic monitoring using real video feeds instead of simulated data. The Jetson Nano can be deployed at each junction to run the YOLO based detection

model locally, reducing latency and enabling faster decisions without heavy cloud dependence. Multiple such nodes can be connected through IoT-based communication for coordinated signal control across junctions. Future enhancements may also include emergency vehicle detection with automatic priority green phases, adaptive control for peak hours and special zones, and integration with smart city infrastructure for centralized monitoring across the entire road network.

V. CONCLUSION

The proposed Dynamic Traffic Signal Allocation System presents an efficient solution to the problem of traffic congestion by combining computer vision for vehicle detection and reinforcement learning for signal timing decisions. The system dynamically adjusts traffic signals based on actual traffic conditions detected from the simulation environment. The use of SUMO provided a controlled environment to test the system under different traffic scenarios, while the dashboard enabled real-time monitoring and analysis. The backend effectively handled data processing, model execution, and communication, whereas the frontend provided clear visualization and user interaction. The hardware demonstration using Raspberry Pi and LEDs further validated the practical applicability of the system. Overall, the system successfully reduces waiting time, improves traffic flow, and demonstrates strong potential for real-world deployment in modern urban traffic management.

ACKNOWLEDGMENT

The authors would like to express sincere gratitude to the project guide and faculty members for their continuous support, guidance, and encouragement throughout the development of this project. Their valuable suggestions and insights helped in improving the quality of the work and overcoming various challenges. The institution is also acknowledged for providing the necessary resources and environment to successfully complete this project. Additionally, the developers of open-source tools, datasets, and platforms including SUMO, YOLOv5, Firebase, and TraCI are acknowledged for making their work publicly available. Finally, thanks are extended to peers and colleagues for their cooperation and support during the project development.

REFERENCES

[1] F. Rasheed, K.-L. A. Yau, R. M. Noor, C. Wu, and Y.-C. Low, “Deep Reinforcement Learning for Traffic Signal Control: A Review,” IEEE Access, vol. 8, pp. 208016–208044, 2020

[2] Z. Wu, S. Wang, C. Ni, and J. Wu, “Adaptive Traffic Signal Timing Optimization Using Deep Reinforcement Learning in Urban Networks,” Artificial Intelligence and Machine Learning Review, Elsevier, 2024

[3] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, “Reinforcement Learningbased Multi-agent System for Network Traffic Signal Control,” IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128–135, 2010.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting,” in Proc. ICLR, 2018.

[5] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li, “CoLight: Learning Network-level Cooperation for Traffic Signal Control,” in Proc. CIKM, 2019, pp. 1913–1922.

[6] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, “MetaLight: Value-Based Meta-Reinforcement Learning for Traffic Signal Control,” in Proc. AAAI, 2020, pp. 1153–1160.

[7] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning,” in Proc. ICML, 2018, pp. 4292– 4301.

[8] X. Ma, Q. Zhang, J. Xu, X. Chen, and S. Zhao, “MAUCE: MultiAgent Upper Confidence Exploration with Coordination Graphs,” in Proc. ICML, 2018.

[9] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, “A Survey and Critique of Multiagent Deep Reinforcement Learning,” in Proc. IJCAI, 2021, pp. 4512–4520.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. CVPR, 2016, pp. 779–788.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks," in Proc. NIPS, 2015, pp. 91–99.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in Proc. ECCV, 2016, pp. 21–37.

[13] T. Che, Y. Author, and Z. Contributor, "Unsupervised Domain Adaptation for Semantic Segmentation in Autonomous Driving: A Survey," Elsevier, 2023.

[14] T. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in Proc. ICLR, 2017.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in Proc. NeurIPS, 2017, pp. 6000–6010.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," arXiv preprint arXiv:1509.02971, 2015.

[17] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in Proc. ICML, 2017.

[18] L. Li, Y. Lv, and F.-Y. Wang, "Traffic Signal Timing via Deep Reinforcement Learning," IEEE/CAA Journal of Automatica Sinica, vol. 3, no. 3, pp. 247–254, 2016.

[19] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intelligent Traffic Signal Control: A Review," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 4, pp. 1–35, 2019.

[20] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in Proc. ECCV, 2020, pp. 213–229.