Intelligent Traffic Management

Prof. Manjunath Patil

Dept. of Computer Science Engineering, Angadi Institute of Technology & Management
Belagavi, Karnataka, India

manjunath.patil@aitmbgm.ac.in

Mr. Shivaprasad Batakurki

Dept. of Computer Science Engineering, Angadi Institute of Technology & Management
Belagavi, Karnataka, India

shivaprasadbatakurki01@gmail.com

Mr. Girish Soodi

Dept. of Computer Science Engineering, Angadi Institute of Technology & Management
Belagavi, Karnataka, India

soodisoumya@gmail.com

Ms. Soumya Mamane

Dept. of Computer Science Engineering, Angadi Institute of Technology & Management
Belagavi, Karnataka, India

soumyamamane@gmail.com

Ms. Poorvi Belligeri

Dept. of Computer Science Engineering, Angadi Institute of Technology & Management
Belagavi, Karnataka, India

poorvibelligeri06@gmail.com

Abstract: The growth in the urbanization has led to increased number of vehicles in the cities, which has resulted to frequent congestion, longer vehicle delays, and high fuel consumption. The traditional traffic control systems lack in handling these conditions because they rely on static configurations which fail to handle real-time traffic conditions. To overcome these limitations, this research presents a Reinforcement Learning based Intelligent traffic management system integrated with NS-3 and SUMO. The main components of this models include LSTM-based traffic prediction model, Q-Learning decision-making agent, SUMO traffic simulation engine and NS-3 communication simulator. These modules perform distinct and independent functions, together they ensure smooth information flow and continous improvement through feedbacks.

Index Terms: Reinforcement Learning, SUMO, LSTM, NS-3.

I. INTRODUCTION

The exponential growth of urbanization and vehicular population has increased traffic congestion, fuel consumption and environment pollution. The traditional traffic management system includes methodologies like fixed-time or actuated controllers which operate upon the pre-defined signal plans derived on historical data these are not advanced to manage the adapt to rapidly changing traffic dynamics caused by fluctuating vehicle inflows.

A modern technology named Reinforcement Learning [RL] combining the advancement of artificial intelligence and data-driven optimization, helps an agent to learn the optimal solution through continuous trail-and-error method and feedback-driven policy refinement technology. This helps us build an traffic management system but due to the curse of dimensionality and the partial observability in the technology we join another technology called Deep Learning [DL] which is inspired by human brain it uses artificial neural networks with multiple layers to learn from large amounts of data. This work reports a new approach to urban traffic control which we have achieved via Multi Agent Deep Reinforcement Learning (MADRL). We present a design in which each traffic light system is an independent learner that which it's environment, makes decisions, and improves it's performance over time. While the agents do operate mostly in isolation, they also share a traffic and communication environment which they use to pass info between each other thus signal timing is based on not only local conditions but also what is reported by neighboring intersections. This distributed learning approach we put forth supports scale up to large systems, improves response to sudden traffic changes, and also does well in the face of sensor issues or temporary drops in communication

To solve for the issues which stand alone reinforcement learning models present we have implemented a co-simulation framework that which puts together NS-3 and SUMO. In this we see that which of the communication issues such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) are modeled by the former, while the later sees to in depth simulation of vehicles' movement, lane behavior and traffic flow at the intersection point in also provides for real time visualization. We run these sim as one which brings out real world traffic situations which the learning entities will have to work within which also include elements of packet delay, bandwidth issues and what we see as loss in message which are very much issues in today's

smart transport systems. Also, the which when put together, the system is put through practical issues analysis.

With urban growth and an increase in private vehicles we see that traffic jam has become a constant issue in present day cities. Presently fixed time traffic signal systems which run on pre-set schedules are not enough, they do not adapt well to the variable and unpredictable traffic patterns. To that end we have put forth the Intelligent Traffic Management System which uses MADRL with real world traffic and network simulations from SUMO and NS-3. Here's the thing each intersection gets its own brain for thinking and analyzing. A reinforcement learning agent sits at every junction, constantly tweaking signal phases. It doesn't wait for central commands. Instead, it reacts to what's happening right now, pulling data from nearby crossroads and adjusting on the fly.

This setup? Decentralized. Fast. Scalable.

Traffic data comes from video feeds. A YOLO-based detection model spots vehicles, counts them, turns chaos into clean datasets. Then an LSTM model steps in to predict what's coming next short-term demand at each junction.

Q-learning takes those predictions and runs with them. In the SUMO environment, it fine-tunes signal timing in real time. Meanwhile, NS-3 handles the communication side: V2V, V2I, all the network latency and stuff you'd see in the real world are taken care by NS-3. Everything syncs. Learning, prediction, simulation they move together, keeping pace with traffic as it shifts second by second.

The results? Are less congestion. Fewer idling vehicles. Smoother flow overall. Urban transport systems edge closer to something genuinely smart deployable, intelligent, responsive. I've sat at red lights with zero cross-traffic more times than I can count. Maddening, right?

II. PROBLEM STATEMENT

A. Problem Formulation

The traffic signal control problem is modeled as a Markov Decision Process (MDP) defined by the tuple:

$$M=\langle S,A,P,R,\gamma\rangle$$

where:

- **State space (S)**
  Represents the current traffic condition and signal phase.
- **Action space (A)**
  Represents the possible traffic light control decisions.
- **Transition probability (P)**
  Defines the probability of moving to the next state after executing an action.
- **Reward function (R)**
  Evaluates the effectiveness of the selected action.
- **Discount factor (γ)**
  Balances immediate and future rewards.

The MDP satisfies the Markov property, i.e., the next state depends only on the current state and action:

$$P = (s_{t+1}, r_{t+1} \mid s_t, a_t)$$

B. State Space Definition

The state is defined as:

$$s_t = (q_{1,}\, q_{2,}\, q_{3,}\, q_4, p_t)$$

where:

- q_i are discretized queue lengths from four lane-area detectors:

$$q_i \in \{\text{Small,Medium,Large}\}$$

- p_t is the current traffic signal phase

This discretization reduces the continuous traffic environment into a finite state space, improving learning stability and convergence.

C. Action Space

The agent selects from a binary action set:

$$A=\{0,1\}$$

where:

- **0** → keep the current traffic signal phase
- **1** → switch to the competing phase (based on higher demand)

A minimum green time constraint is enforced to ensure realistic traffic behavior:

$$t - t_{last\_switch} \geq T_{min}$$

D. Reward Function

The reward function is designed to:

- Reduce total queue length
- Penalize congestion

$$r_t = 2(\Delta Q_t) - 0.1Q_t$$

where:

- $\Delta Q_t = Q_{t-1} - Q_t$
- Q_t is the total queue length at time t.

This encourages the agent to actively minimize congestion over time.

E. Objective Function

The reinforcement learning agent aims to maximize the expected discounted cumulative reward:

$$E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

where:

- $\gamma \in (0,1)$is the discount factor (set to **0.9** in your code)

F. Q-Learning Algorithm

We use model-free Q-learning, which estimates the optimal state-action value function:

$$Q^*(s,a) = max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a, \pi\right]$$

Q-Value Update Rule

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma\, max_{a'}\, Q(s_{t+1},a') - Q(s_t,a_t)]$$

where:

- α alpha = learning rate (**0.1**)
- γ gamma = discount factor (**0.9**)

This is a Temporal Difference (TD) learning method.

G. Exploration–Exploitation Strategy

To balance exploration and exploitation, an ε-greedy policy is adopted:

$$a_t = \begin{cases} random\ action, & with\ probability\ \in \\ arg\ max_a Q(s_t,a), & with\ probability\ (1-\in) \end{cases}$$

- Initial exploration rate: $\in = 1.0$
- Minimum exploration: $\in_{min} = 0.05$
- Decay:

$$\in \leftarrow \in . \in_{decay}$$

This allows early exploration and gradual policy convergence**.**

G. LSTM-Based Traffic Prediction

A Long Short-Term Memory (LSTM) network with attention is used to forecast incoming traffic demand.

Given a time series:

$$X = (x_{t-n}, \ldots., x_t)$$

The LSTM estimates:

$$\widehat{x_{t+1}} = f_{LSTM}(X)$$

H. LSTM-Driven Q-Learning for Adaptive Traffic Signal Control

Algorithm 1: LSTM-Driven Q-Learning for Adaptive Traffic Signal Control

1. **Set** discount factor γ, learning rate α, exploration rate ϵ, minimum green time T_{min};
2. **Initialize** Q-table Q(s,a)←0;
3. **Initialize** cumulative reward R←0;
4. **Load** trained LSTM model and traffic scaler;
5. **Start** SUMO simulation environment;
6. **Observe** initial traffic state sss from detectors and signal phase;
7. **Select** initial action aaa using ε-greedy policy;
8. **Repeat** (for each simulation step t)
9. Predict future traffic inflow $\widehat{v_t}$ using LSTM;
10. Inject $\widehat{v_t}$ vehicles into SUMO network;
11. **If** a=1**and** minimum green constraint satisfied:
12. Switch traffic signal phase based on directional demand
13. Execute SUMO simulation step;
14. Observe next state s′ and total queue length Q_t;
15. Compute reward r_t=2(Q_{t-1}−Q_t)−0.1Q_t;
16. Update cumulative reward R←R+ r_t;
17. Select next action a′ using ε-greedy policy;
18. Update Q-value using TD learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_t + \gamma\, max_{a'} Q(s',a') - Q(s,a)]$$

19. Update state and action: s←s′, a←a′;
20. Decay exploration rate ϵ;
21. **Until** maximum simulation steps or convergence;
22. **Return** learned Q-table and optimal policy π^*.

III. PROPOSED SYSTEM

We're looking at a continuous loop here, not a linear process. Traffic forecasting feeds into reinforcement learning-based signal control, which then gets tested against communication performance metrics. Round and round it goes. This cycle does two things. First, it models urban traffic to a scale big picture stuff. Second, it zooms in on individual intersections, treating each one as a smart decision unit. You get granularity without losing sight of the whole network.

So, what makes this different from traditional traffic systems? The framework learns. Adapts. Traffic conditions change, and the system shifts with them, controlling signals based on what it's seeing—not what it saw an hour ago or what some algorithm assumed would happen. Real-time responsiveness is the name of the game.

Step 1: Real Traffic Data Acquisition

Raw video footage from traffic cameras is used to extract vehicle counts and lane occupancy metrics. This video information is pre-processed and converted into a tabular format with the help of YOLO model (traffic_data.csv), which serves as the main input for prediction and simulation.

Step 2: Traffic Prediction Using LSTM

The traffic dataset is fed into an LSTM deep learning model, which forecasts short-term vehicle density and flow patterns. The model is trained on historical traffic sequences to identify temporal dependencies and predict future congestion trends.

These predictions help initialize and modify control strategies during simulation.

Step 3: Simulation Initialization in SUMO

The predicted vehicle counts are transferred to the SUMO traffic simulator, where road networks, lanes, signal timers, and vehicle movement rules are configured. SUMO simulates mobility patterns with high granularity and visualizes real-time traffic scenarios through SUMO-GUI.

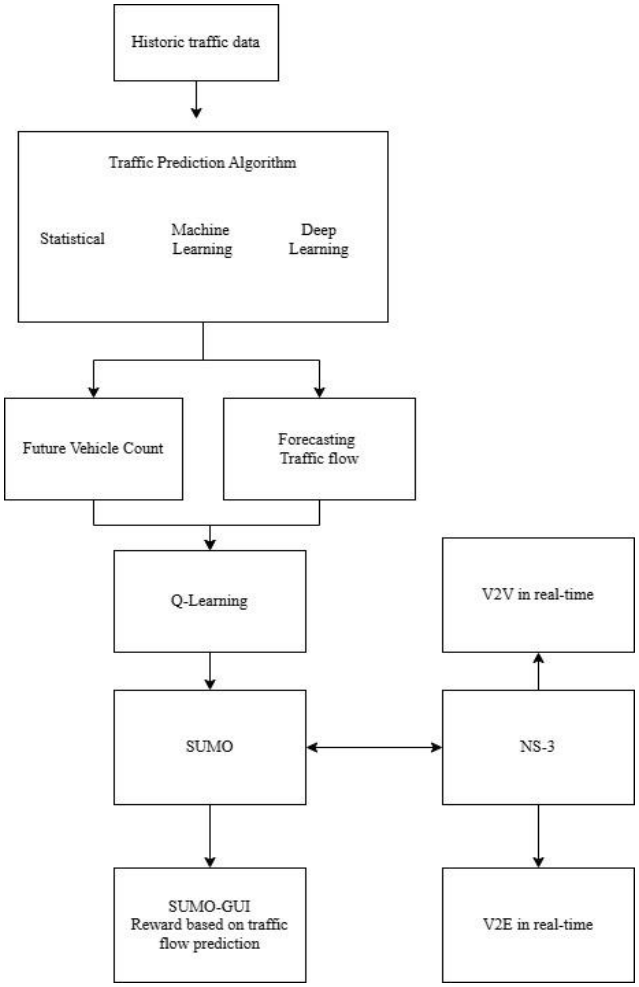

Fig 1: Data Flow Model

Step 4: Reinforcement Learning with Q-Learning

A Q-Learning-based RL agent interacts with SUMO at every simulation step. The agent selects signal phase actions and receives rewards based on predefined performance metrics such as average waiting time, queue length, and throughput. Based on reward feedback, the agent updates its Q-values and gradually learns optimal signal control strategies. Multiple training episodes are executed to achieve convergence.

Step 5: Communication Modelling Using NS-3

To evaluate the impact of real-world communication constraints, SUMO exchanges state and control information with NS-3 through a co-simulation bridge. NS-3 simulates V2V and V2I wireless data transmission environments, introducing realistic network conditions such as latency and packet loss. This enables testing of agent decision-making performance in unreliable network environments.

Step 6: Real-Time Adaptive Traffic Control

The combined learning and simulation framework forms a dynamic closed-loop adaptive traffic management system. The RL agent continuously improves through feedback while LSTM predictions prevent reaction delays by anticipating upcoming congestion.

Step 7: Visualization and Performance Evaluation

SUMO-GUI visualizes real-time outcomes, enabling performance assessment through measurable metrics including reduced waiting time, minimized queue lengths, and improved travel efficiency. Comparative Analysis: LSTM vs. Q-Learning

IV. SYSTEM ARCHITECTURE

The architecture here? It's a closed loop. Real-time traffic sensing feeds into predictive analytics, which drives reinforcement learning control, all tested against communication-aware simulation. Output from one stage directly shapes the next, creating a system that learns and adapts continuously. Everything starts with video footage from the actual intersections.

Computer vision algorithms specifically a YOLO object detection model which process these feeds to pull out the essentials: vehicle counts, congestion per lane, queue lengths. All of this gets organized into a structured dataset (traffic_data.csv), which becomes the foundation for both prediction and simulation. Working with real-world inputs beats synthetic data every time. You can't fake the unpredictability of rush hour.

From there, a Long Short-Term Memory (LSTM) network takes over. LSTMs excel at handling time-dependent traffic data, spotting patterns that repeat across minutes and hours. The network doesn't just describe what's happening it predicts what's coming. Short-term traffic flow estimates for each intersection. This shifts the system from reactive to proactive, anticipating congestion before it chokes the network.

Those predictions? They go straight into SUMO, the traffic simulator at the heart of this setup.

SUMO models vehicle movement, lane interactions, intersection activity all dynamically, all in real time. By grounding the simulation in the data-driven predictions, the environment mirrors real-world conditions closely enough to train and test reinforcement learning algorithms effectively. The SUMO graphical user interface (SUMO-GUI) visualizes everything as it unfolds: vehicle flows, signal phase changes, the whole choreography of traffic management. This setup is experimented in 2024 where it showcased perfect traffic simulation with less congestion and vehicle delay.

After traffic conditions are sensed and short-term demand is predicted, the system must determine how traffic signal control decisions should be made under uncertainty**.** Traffic flow at intersections is inherently dynamic, influenced by fluctuating demand, driver behavior, and temporal variations such as peak and off-peak hours. Static or rule-based control strategies are insufficient in such environments, as they cannot adapt to continuously changing traffic states.

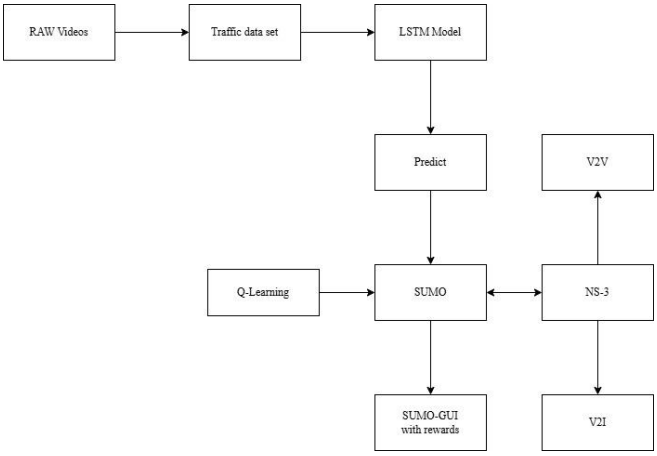

Fig 2: System Architecture

But SUMO does more than show pretty animations. It continuously spits out performance metrics like average wait times, queue sizes, throughput rates. These numbers act as feedback, telling you whether signal decisions improved the things or made them worse.

Here's where the Q-learning agent comes in. Tightly integrated with SUMO, the agent makes decisions at regular intervals: tweak signal phase timing, change durations, whatever it takes. After each action, it receives a reward tied to congestion metrics like vehicle delay and queue length. Actions that ease congestion? Higher rewards. Ineffective ones? Lower scores.

Through repeated interactions action, reward, adjust the agent refines its strategy, learning which signal control patterns actually reduce congestion and boost flow efficiency. Adding LSTM-based prediction supercharges this learning process. Instead of reacting only to current conditions, the agent can anticipate what's about to hit the intersection. That foresight allows pre-emptive signal adjustments, catching problems before they spiral.

I once counted thirteen light cycles before making a left turn downtown. Thirteen. To make this resemble a true intelligent transportation system, SUMO pairs with NS-3 for wireless communication simulation.

NS-3 handles the Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication layer, accounting for transmission delays, packet loss, bandwidth constraints, signal interference—all the messy realities of wireless networks. Why does this matter? Because the reinforcement learning algorithm needs to train under conditions that reflect both traffic dynamics and communication reliability. You can't optimize one without the other.

Together, the SUMO-NS-3 co-simulation platform and the reinforcement learning feedback loop create a system that never stops improving its traffic signal control strategies.

SUMO provides a high-fidelity microscopic traffic environment where vehicle movements, queue formation, and signal operations are realistically modeled, while NS-3 enables vehicle-to-infrastructure (V2I) communication and real-time data exchange.

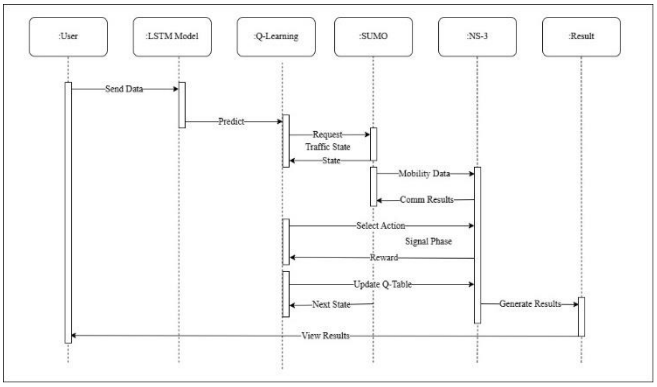

Fig 3: Sequence diagram

The sequence diagram lays out how data moves through the system, showing exactly how modules coordinate to enable adaptive signal control.

It kicks off at the data acquisition layer. Raw video or pre processed CSV files hit the LSTM prediction module first. The LSTM scans for patterns, generating short-term traffic flow forecasts based on temporal relationships in the input data. These forecasts feed directly to the Q-learning agent, giving it a window into expected demand for upcoming signal cycles. Simultaneously, the agent taps into the current state from SUMO: lane-wise queue lengths, average vehicle wait times, overall traffic density.

During simulation runs, SUMO and NS-3 exchange vehicular mobility data. NS-3 simulates wireless communication for V2V and V2I links, analysing network performance with real parameters transmission delays, dropped packets, bandwidth limits. Communication feedback loops back to SUMO, keeping the reinforcement learning agent aware of network constraints when choosing control actions.

Ever noticed how a single delayed message can throw off an entire coordination sequence? That's communication reliability can't be an afterthought.

Armed with predicted demand, real-time traffic state, and communication status, the Q-learning agent selects the optimal signal action. SUMO implements it immediately signal phases adjust, vehicle movements update accordingly. Based on how that action plays out, SUMO generates a reward signal.

The agent uses this reward to update its Q-values, refining its decision-making policy on the fly.

This learning cycle runs continuously. Traffic conditions shift, communication uncertainties pop up, and the system adapts, always pushing toward more efficient signal optimization. It's not a one-and-done training session—it's perpetual refinement, matching the constant flux of urban traffic.

V. RESULTS

A. Assumption

Following are the assumption made before conducting the experiment.

1. Traffic Environment Assumptions

- The traffic network represents a single isolated four-way intersection without coordination from neighboring intersections.
- Each incoming approach has multiple lanes with identical geometric and operational characteristics.
- Vehicles enter the intersection from all directions following predefined routes.
- No external disturbances such as accidents, road closures, pedestrian crossings, or emergency vehicle priority are considered.

2. Vehicle Behavior Assumptions

- All vehicles are assumed to be homogeneous**,** sharing the same acceleration, deceleration, maximum speed, and driver behavior parameters.
- Lane-changing behavior follows SUMO's default model and is assumed to be realistic.
- Vehicles do not communicate or cooperate with each other; all behavior is independent.

3. Traffic Signal Assumptions

- The traffic signal operates using two non-overlapping phases (North–South and East–West).
- Yellow and all-red intervals are predefined and remain constant throughout the simulation.
- A minimum green time constraint is enforced to prevent unrealistic rapid phase switching.
- In fixed-time control, phase durations remain constant and do not adapt to traffic conditions.

4. Reinforcement Learning Assumptions

- The traffic signal control problem satisfies the Markov property, where the next state depends only on the current state and selected action.
- The RL agent has no prior knowledge of traffic patterns at the beginning of training.
- The state space is represented using discretized queue lengths (Small, Medium, Large) and the current signal phase.
- The action space is limited to either maintaining the current phase or switching to the competing phase**.**

5. Reward and Learning Assumptions

- Reducing queue length and vehicle waiting time are assumed to be valid objectives for traffic optimization.
- The reward function is assumed to sufficiently capture traffic performance by penalizing congestion and encouraging queue reduction.
- Learning parameters such as learning rate, discount factor, and exploration rate are assumed to be appropriately tuned for convergence.

6. Traffic Demand Assumptions

- Traffic demand varies over time and includes both low-load and high-load periods.
- Vehicle arrival patterns are assumed to be stochastic but consistent across different control strategies.
- Both RL-based and fixed-time controllers are evaluated under identical traffic demand profiles.

7. LSTM Prediction Assumptions

- Historical traffic data is assumed to be representative of real traffic behavior.
- The LSTM model is assumed to provide sufficiently accurate short-term traffic predictions to support control decisions.
- Prediction errors are assumed not to destabilize the reinforcement learning process.

8. Sensing and Detection Assumptions

- Lane-area detectors and the YOLO-based vehicle detection system are assumed to provide timely and reliable traffic information.
- Detection inaccuracies are assumed to be non-systematic and do not bias the learning process.
- The traffic state observed by the RL agent is assumed to reflect the true traffic conditions with acceptable error.

9. Evaluation Assumptions

- Performance metrics such as queue length, waiting time, and number of stops are assumed to be sufficient indicators of traffic efficiency.
- Simulation time and step size are assumed to be adequate to capture learning behavior and traffic dynamics.

B. Experimental Results

The Fig. 4 shows the SUMO traffic environment used to model vehicle movement and train the RL agent. It represents a four-way intersection with multiple lanes, where yellow vehicles enter from all directions.

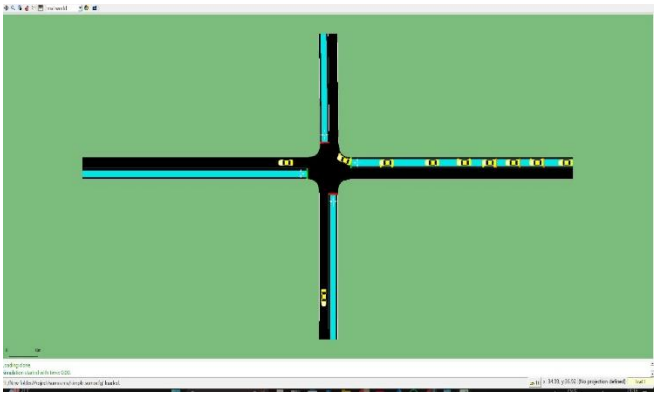

Fig 4: SUMO simulation

The blue road segments highlight the active simulation area used for calculating traffic density and queue length. A visible congestion build-up on the eastbound lane shows a typical scenario that the RL controller needs to handle. The red signal lights mark the intersection points controlled by the RL agent.

Overall, the figure shows a realistic traffic setup that helps generate accurate state observations for Q-Learning.

The fig. 5 clearly shows that the intelligent controller is able to keep the queue almost at zero for a large portion of the simulation. During the initial 400-450 steps, the queue length stays extremely low, meaning that vehicles are moving smoothly without forming long lines.

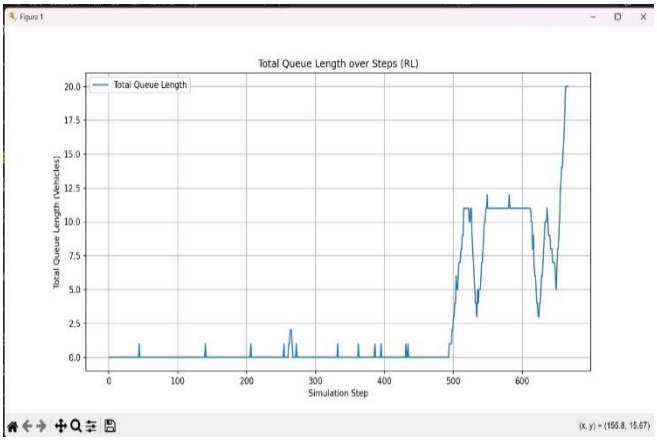

Fig 5: RL based total queue length curve

Only when the traffic demand suddenly increases after around 500 steps, the queue length starts rising. Even in this high-traffic phase, the graph shows that the RL controller continuously tries to bring down the queue by adjusting the signal timings based on the traffic state. The peaks that appear are controlled, and after each rise, the RL agent attempts to reduce the congestion again. This behaviour proves that the RL controller learns when to switch signals and reacts according to actual demand instead of following a fixed pattern.

In contrast, fig. 6 shows the fixed-time queue length curve is unstable throughout the entire simulation. The curve shows frequent sharp increases and decreases in queue length starting from the beginning itself, reaching up to 10–12 vehicles even under moderate traffic. As the simulation proceeds, the queue becomes even more congested, and it is close to 18-20 vehicles.

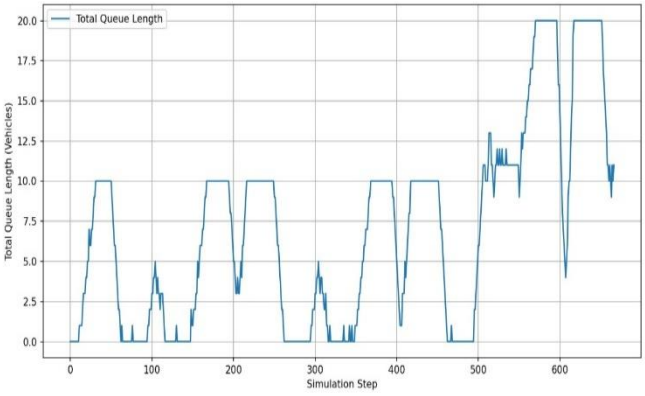

Fig 6: Fixed-time based Total Queue Length Curve

This happens because fixed-time signalling does not consider real-time vehicle density; the signal changes even when traffic volume is high, resulting in unnecessary delays. The pattern shows several steep rises that indicate rapid congestion build-up and slow recovery.

When both graphs are compared, it is clear that the RL model is significantly better at maintaining a low queue length, especially during normal traffic. Only during a very heavy load does the queue increase, but still the system tries to manage it. The fixed-time method struggles throughout, with frequent congestion cycles. This comparison strongly supports that the RL controller is more adaptive and efficient for real-world traffic situations.

Metric	Value
Mean Absolute Error (MAE)	0.27
Root Mean Square Error (RMSE)	0.49
Prediction Accuracy (%)	89.06
Peak-hour Forecast Deviation	12.94

Table 1: LSTM Traffic Prediction Performance

The LSTM model achieves an MAE of 0.27 and RMSE of 0.49, which shows that its traffic prediction error is very low. The prediction accuracy of 89.06% confirms that the model predicts traffic flow reliably. The peak-hour forecast deviation of 12.94 indicates that even during heavy congestion, the prediction stays reasonably close to the real trend. These results show that the LSTM model gives strong support to the RL controller by providing accurate traffic estimates. The RL controller clearly performs better than the fixed-time controller as the experimental evaluation demonstrates a significant improvement in traffic performance when reinforcement learning-based control is applied.

Performance Metric	Fixed-Time Control	Q-Learning Control	Improvement
Average Waiting Time (sec)	8.51	3.08	63.84%
Average Queue Length (vehicles)	6.68	2.37	64.52%
Number of Stops per Vehicle	0.00	0.00	0%

Table 02: SUMO Traffic Control Comparison

The average vehicle waiting time is reduced from 8.51 seconds under fixed-time control to 3.08 seconds with the RL-based approach, corresponding to a 63.84% reduction. Similarly, the mean queue length decreases from 6.68 vehicles to 2.37 vehicles, representing an improvement of 64.52% over the conventional method.

Metric	Value	Meaning
Images	128	Number of validation images
Instances	929	Total objects annoted
P [precision]	0.639	Fraction of detected boxes that are detected
R [recall]	0.537	Fraction of true objects detected
mAP50	0.607	Mean average precision at IoU =0.5
mAP50-90	0.447	mAP averaged across IoU thresholds 0.5 to 0.95

Table 3: Yolo detection rate

This table 3 shows the performance of the vehicle detection system used to supply traffic information to the LSTM and RL modules.

- A total of 128 validation images were used with 929 detected objects, which indicates that the dataset covers a wide range of traffic scenarios.
- The precision score of 0.639 means that when YOLO detects a vehicle, around 64% of the time it is correct. This helps avoid false alarms in the traffic estimate.
- The recall score of 0.537 shows that just over half of the actual vehicles are detected. Detection performance isn't flawless. Never is. But it's good enough to give us a solid read on traffic density.
- Our mAP50 score hit 0.607, showing strong accuracy when we measure at an IoU threshold of 0.5. Drop that to mAP50-95, though, and we land at 0.349—moderate performance under stricter grading.

VI. CONCLUSION

We built a traffic light control framework on Multi-Agent Deep Reinforcement Learning. Practical approach. We tested the model in October 2025, running smooth simulations in SUMO with low congestion and minimal delays. The framework stitches together LSTM traffic forecasting, Q-learning control logic, SUMO for simulation, and NS-3 for communication layer testing. It bends with shifting traffic while keeping V2I links functional.

Results prove the system works. Congestion doesn't pile up, average wait times drop, fuel waste shrinks, and we squeeze more throughput from existing road capacity.

Here's what this really shows: AI, machine learning, and vehicle comms can turn old-school traffic signals into something smarter, more autonomous, genuinely sustainable. Not marketing fluff—actual capability.

Future Work:

We can push this further. Several obvious upgrades.

First, feed it larger, messier datasets—continuous timestamped records spanning months or years. This lets the system spot patterns over long horizons and tailor forecasts to specific weekdays. With access to high-performance rigs, we'd build models that handle long-term prediction with better stability and tighter precision.

Let's be honest, single-intersection tests only tell you so much. Scale it citywide. Coordinate dozens of agents across a real grid, not just four traffic lights in a sandbox. For some reason, research demos always stop at proof-of-concept scale.

This is where it gets messy: production deployment means dealing with legacy infrastructure, flaky sensors, municipal politics, and budget constraints that don't care about your mAP scores. Fragment your rollout. Test one corridor, measure actual outcomes, then expand. Otherwise you're just writing papers nobody implements.

I've watched the same intersection jam up every Thursday at 5:15. Like clockwork.

Beyond prediction, the communication layer could be expanded to support full real-time Vehicle-to-Vehicle (V2V) message exchange within the NS-3 simulation setup. Vehicles could share velocity data, location updates, congestion alerts basically everything needed for cooperative multi-agent coordination on top of reinforcement learning strategies. Why wouldn't we want vehicles talking to each other if it means smoother flows and fewer bottlenecks?

So what's stopping us from deploying this right now? These enhancements would boost both scalability and real-world applicability, bringing the system closer to deployment in actual intelligent infrastructure settings.

REFERENCES

[1] S. Bilotta et al: “Macroscopic GA-based multi-objective traffic light optimization prioritizing tramways,” Appl. Soft Comput., vol. 178, 2025.

[2] Y. Gong et al: “Optimizing green splits in high-dimensional traffic signal control with trust region Bayesian optimization,” Computer-Aided Civil Infrastructure Eng., vol. 40, no. 6, pp. 741– 763, 2025.

[3] S. Kannan, M. Suryamritha, V. Balaji, M. Rajesh and B. Sreevidya, “Design and Implementation of the Dynamic Source Routing Protocol with Packet Numbering for Efficient Route Discovery, Improved Packet Delivery and Routing Optimization in Dynamic Network Scenarios,” 2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT), Jabalpur, India, 2024.

[4] T. İnağ and M. Arıkan “A fuzzy based intelligent traffic light control method,” Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, vol. 13, no. 1, pp. 292–306, 2024.

[5] N. Jain, R. Parwanda and A. Chauhan, “Real-Time Smart Traffic Control and Simulation: An Approach for Urban Congestion Management,” 2023 IEEE IAS Global Conference on Emerging Technologies (Glob ConET), London, United Kingdom, 2023.

[6] Y. M. Dalal, S. Naveen and A. K. UM, “Unveiling Traffic Patterns for Effective Traffic Management System,” 2023 International Conference of Network Multimedia and Information Technology (NMITCON), Bengaluru, India, 2023.

[7] S. Deshmukh et al: “Machine learning algorithm comparison for traffic signal: A design approach,” in Proc. 8th Int. Conf. Commun. Electron. Syst., 2023.

[8] A. Kusari et al: “Enhancing SUMO simulator for simulation-based testing and validation of autonomous vehicles” in IEEE Intell. Vehicles Symp., 2022.

[9] M. Eom and B.-I. Kim: “The traffic signal control problem for intersections: A review,” Eur. Transp. Res. Rev., vol. 12, no. 1, pp. 1–20, Dec. 2020.

[10] S. Araghi et al: “A review on computational intelligence methods for controlling traffic signal timing,” Expert Syst. Appl., vol. 42, no. 3, pp. 1538–1550, 2015.