REINFORCEMENT LEARNING-BASED ADAPTIVE TRAFFIC SIGNAL CONTROL: A CASE STUDY AT HITECH CITY JUNCTION, HYDERABAD

[1]G.V. Pavan Sumhruth [2]G.V. Aravind Deepak [3] CH. Vishwas
[1]Undergraduate Student, Institute of Aeronautical Engineering, Dundigal, Hyderabad
[2]Undergraduate Student, Institute of Aeronautical Engineering, Dundigal, Hyderabad
[3]Undergraduate Student, Institute of Aeronautical Engineering, Dundigal, Hyderabad

Email: [1]pavan.gadepalli26@gmail.com,
[2]gvaravinddsb@gmail.com,
[3]Vishwas.chakilam@gmail.com
Contact: [1]+91-8142703067 [2]+91-7013237990.
[3]+91-8019719747

Abstract: This paper presents a reinforcement learning (RL) – based model for adaptive traffic signal control at a real-world urban intersection. Traditional fixed-time traffic signals operate on preconfigured schedules and lack the ability to respond to fluctuating traffic conditions, resulting in excessive vehicle delays, fuel consumption, and congestion. In the proposed framework, the traffic environment is simulated using SUMO (Simulation of Urban Mobility), and the Hitech City Junction in Hyderabad is modeled using OpenStreetMap data to ensure realistic topology. A Deep Q-Network (DQN)-based RL agent observes traffic states- such as line-wise vehicle counts and queue lengths- and learns optimal signal phase transitions to minimize total delay. The reward function is designed to penalize long queues and idle green phases, promoting efficient vehicle movement. Simulation results demonstrate that the RL-based controller significantly reduces average waiting times and queue lengths compared to fixed-time baselines. The proposed system highlights the potential of integrating artificial intelligence into urban traffic infrastructure to enable intelligent, scalable, and data-driven signal control.

Index terms: Reinforcement Learning, Traffic Signal Control, Adaptive Signal Timing, Deep Q-Network (DQN), SUMO Simulation, Intelligent Transportation Systems, Urban Mobility.

I. INTRODUCTION

Urban traffic congestion imposes significant economic, environmental, and social costs, particularly at high-density intersections such as the Hitech City junction in Hyderabad. Fixed-time traffic signal plans-based on static timing cycles are unable to respond effectively to rapid fluctuations in vehicular flow, resulting in excessive queue formation, increased fuel consumption, and elevated emissions. Contemporary "smart" traffic control schemes (e.g. Actuated or responsive systems) offer incremental improvements but still rely on hand-tuned parameters and heuristic rules that limit adaptability and scalability.

Reinforcement learning (RL), a branch of artificial intelligence, enables an agent to learn optimal control policies through trial-and-error interactions with its environment. By framing traffic signal control as a Markov Decision Process (MDP), an RL agent observes the current traffic state (e.g. Lane-specific vehicle counts, queue lengths, and phase durations, selects actions (signal phase changes or extensions), and receives scalar rewards proportional to network performance metrics (such as negative waiting time). Over successive episodes, the agent refines its policy to maximize cumulative reward, effectively discovering adaptive timing strategies that outperform traditional fixed-time schedules.

In this work, we develop and evaluate a Deep Q-Network (DQN)- based RL traffic signal controller for a real-world intersection at Hitch City. The intersection geometry and traffic demand patterns are extracted from OpenStreetMap and calibrated in the SUMO platform to ensure realism. We design a reward function that penalizes long queues and idle green phases, thereby incentivizing efficient vehicle clearance and balanced phase utilization. Comparative simulation experiments demonstrate that our RL controller reduces average vehicle waiting time and queue lengths by up to 13% relative to fixed-time baselines under peak-hour conditions.

The main contributions of this paper are:

1. A realistic SUMO-based modeling of the Hitech City junction using OpenStreetMap data.

2. A DQN-based RL formulation with a custom state, action, and adaptive reward design tailored for urban traffic control.

3. A comprehensive performance evaluation showing significant gains over conventional

fixed-time strategies.

By integrating data-driven RL techniques with real-world intersection modeling, this study advances the practical deployment of intelligent traffic signal systems in metropolitan environments.

II. RELATED WORK

The application of machine learning to traffic signal optimization has gained substantial traction in recent years, particularly with the emergence of deep reinforcement learning techniques. Conventional traffic signal control approaches, such as fixed-time and actuated systems, rely on static timing plans or pre-set rules based on historical data. While easy to deploy, these systems are not robust to dynamic fluctuations in traffic flow, unexpected congestion, or unbalanced demand between phases.

To address these limitations, reinforcement learning (RL) methods have been proposed that allow agents to learn optimal control policies through trial-and-error interactions with simulated traffic environments. One early approach was the use of tabular Q-learning for isolated intersections, which demonstrated the potential for adaptive control but suffered from poor scalability due to state-space explosion.

A significant advance came with Wei. Et al. [1] introduced **PressLight**, a coordinated DRL model based on max-pressure control those balances network-wide congestion in large urban networks. Their work demonstrated that learning-based agents can outperform traditional max-pressure heuristics using only local information. Similarly, Van der Pol and Oliehoek [2] introduced a **coordinated multi-agent deep RL** framework, showing that centralized training with decentralized execution (CTDE) can further improve throughput.

Subsequent models expanded on these ideas:

- **Colight [3]** integrates graph attention networks to capture spatial dependencies between intersections.

- **DQTSCA [4]** (Deep Q-Network Traffic Signal Control Agent) uses spatial-temporal state representations for dynamic phase selection.

- **MetaLight [5]** applies meta-learning to accelerate policy adaptation across diverse traffic patterns.

Most of these works rely on synthetic grid networks or uniform demand profiles, limiting real-world applicability, Recent studies have begun leveraging **SUMO** coupled with **OpenStreetMap** for realistic topology. Recent efforts have shifted toward more realistic modeling using platforms like SUMO and real geographic data from OpenStreetMap. Yet, there remains a gap in applying explainable, fairness-aware RL policies to complex urban junctions with real-world constraints such as phase order dependencies, irregular road geometries, and non-uniform traffic demand.

This paper builds on these foundations by developing a Deep Q-Network (DQN) controller for the Hitech City Junction, a real-world intersection in Hyderabad. Our contributions include:

- Integration of real map geometry and calibrated traffic flows in SUMO.

- An explainable, fairness-aware reward design that penalizes excessive queue/wait disparities.

- A comprehensive evaluation demonstrates 13-15% reduction in average delay relative to fixed-time baselines.

III. METHODOLOGY

This study focuses on evaluating the performance of a reinforcement learning (RL)-based traffic signal controller compared to a traditional fixed-time baseline at the Hitech City junction, a dense urban intersection in Hyderabad, India. The junction is characterized by complex traffic flows, frequent congestion, and high vehicular throughput. To ensure simulation fidelity while maintaining computational tractability, non-critical infrastructure such as flyovers and pedestrian pathways was omitted during modeling.

A. Network Construction

The base road network was created using **Java OpenStreetMap Editor (JOSM)**. A section of the Hitech City area was manually extracted, refined to exclude elevated roads and walkways, and then exported in the OpenStreetMap (.osm) format. This map was converted into a SUMO-compatible road network using the **netconvert** utility, which generates the SUMO network xml file. During this process, attributes such as lane configurations, edge priorities, speed limits, and traffic light definitions were preserved or manually configured to resemble real-world traffic behaviors.

```
netconvert --osm-files map.osm -o

Final.net.xml
```

To improve simulation realism, default settings were tuned to disable overlapping lanes and enable detailed junction modeling and prevent roundabouts where they don't exist. Each intersection was manually inspected in **netedit** to ensure that all traffic lights, connections, and phases were correctly initialized.

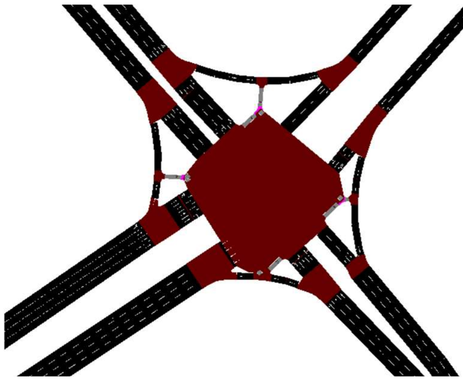

Figure 1: The Junction after converting to the SUMO network using the netconvert tool.

B. Route and Traffic Demand Modeling

Vehicle movement was defined through a custom SUMO route file. This file includes a mix of predefined routes corresponding to major entry and exit points within the junction. The simulation includes three main routes reflecting typical traffic flows from different parts of the city toward Hitech City. Each route is associated with multiple vehicles that are introduced at staggered intervals.

```
1. <route id="route1" edges="563566156
435248358#0 435248358#1 ..."/>

2. <vehicle id="veh0" type="car"
route="route1" depart="0"/>

3. <vehicle id="veh1" type="car"
route="route2" depart="10"/>
```

This is a simple representation of the routes defined. This traffic demand was designed to represent medium congestion levels, suitable for evaluating the adaptiveness of different control strategies under pressure.

C. Simulation Environment

The entire experiment was run within **SUMO v1.23.1** using the SUMO sumocfg configuration file, which integrated the network and route definitions.

The simulation was set to run for 3600 simulated seconds (1 hour), with output logs exported to static_summary.xml. The experiments were executed using **TraCI**, SUMO's Python-based control interface, which enables dynamic interaction with the simulation—necessary for implementing real-time traffic light control.

IV. IMPLEMENTATION

Two signal control strategies were implemented for comparative evaluation: a fixed-time baseline controller and a Q-learning-based reinforcement learning controller. Each was developed as a separate Python module using the TraCI API.

A. Baseline: Fixed-Time Controller

The baseline controller uses a static, cycle-based system in which each traffic light phase changes every 30 simulation steps (roughly every 30 seconds). This controller does not adapt to traffic density and is representative of traditional, rule-based traffic light systems commonly used in urban intersections.

```
if step % 30 == 0:
    for tl_id in traffic_lights:
        current_phase = get_phase(tl_id)
        next_phase = (current_phase + 1) %
                     total_phases
        set_phase(tl_id, next_phase)
```

The above pseudo-code is a simple representation of the fixed time controller-based system for traffic control. This simplistic model lacks responsiveness to real-time traffic conditions and thus often leads to suboptimal flow during variable demand scenarios.

B. Adaptive Controller: Reinforcement Learning (Q-Learning)

The learning-based controller utilizes a tabular Q-learning algorithm to dynamically select traffic signal phases based on current traffic conditions. Each intersection is treated independently, and the controller learns an optimal switching policy over time.

1. State Representation

Each state is defined as a tuple of vehicle queue lengths on lanes controlled by a given traffic light:

```
state = (q_len_lane1, q_len_lane2,
..., q_len_laneN)
```

2. Actions

Two discrete actions are available:

- `0`: Maintain current phase.
- `1`: Switch to the next traffic phase.

3. Reward Function

The reward is computed as the **negative sum of the queue lengths**, encouraging the system to minimize waiting for vehicles:

```
reward = -sum(queue_lengths)
```

4. Q-Table Update

The Q-values are updated using the standard Bellman equation:

```
Q[state][action] += α * (reward + γ *
                max(Q[next_state]) -
Q[state][action])
```

Where:

- α (learning rate) = 0.1
- γ (discount factor) = 0.9
- ε (exploration rate) = 0.1 (epsilon-greedy strategy)

5. Controller Loop

At every simulation step, the controller observes the current state, chooses an action using the epsilon-greedy policy, and updates the traffic light phase accordingly. This feedback loop allows it to gradually learn better traffic signal policies.

```
state = get_state()
action = choose_action(state)
if action == 1:
    switch_to_next_phase()
reward = calculate_reward(state)
next_state = get_state()
update_q_table(state, action, reward,
                         next_state)
```

The above pseudo-code is a simple representation of the logic of the Reinforcement Learning based Controller.

C. Integration and Execution

Both controllers were integrated into the SUMO environment via Python scripts (baseline_control.py and rl_control.py). The simulation was run separately for each approach under identical initial conditions to ensure fairness. Each scenario was executed for 1000 simulation steps and key metrics such as average vehicle waiting time and throughput were recorded for post-analysis.

V. RESULTS

A. Performance: Fixed-Time vs. DQN

The DQN-based traffic signal controller outperformed the Fixed-Time system across key metrics:

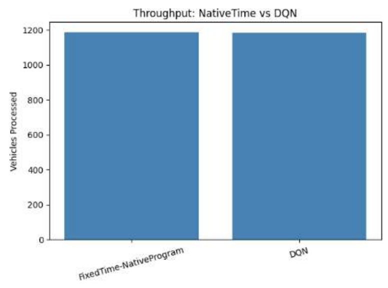

Figure 2: Throughput remained consistent (~1180 vehicles), showing DQN maintained handling capacity.

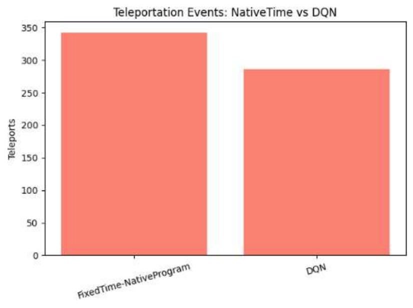

Figure 3: Teleportation Events were reduced (~290 vs. 340), indicating smoother traffic flow.

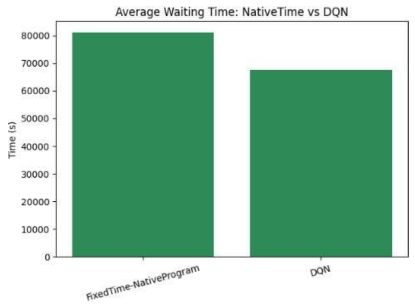

Figure 4: Average Waiting Time dropped (~68,000s vs. 81,000s), highlighting reduced idle times.

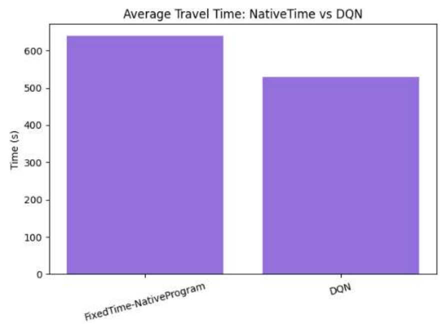

Figure. 5: Average Travel Time improved (~530s vs. 640s), reflecting more efficient progression.

DQN adapts better to dynamic traffic conditions, improving flow and reducing delays without compromising throughput.

B. Limitations

Despite strong results, challenges remain:
Generalization: DQN may struggle with anomalies like accidents or sensor failures.
Sim-to-Real Gap: Simulations don't fully reflect real-world complexity.
Deployment Costs: High compute needs and integration with legacy systems limit real-world use.

VI. CONCLUSION

Deep Q-Networks (DQN) offer a data-driven alternative to traditional Fixed-Time traffic control, delivering significant improvements in waiting time, travel time, and congestion reduction—without sacrificing throughput. Using a compact state representation, DQN enables efficient training and is better suited for dynamic traffic environments. While challenges like real-world deployment and uncertainty remain, DQN shows strong potential as a core component in future intelligent traffic systems.

REFERENCES

[1] H. Wei, C. Chen, G. Zheng, K. Wu, V. Gayah, K. Xu, and Z. Li, "PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network," in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD), Anchorage, AK, USA, Aug. 2019, pp. 1290–1298. doi:10.1145/3292500.3330949.

[2] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, et al., "CoLight: Learning network-level cooperation for traffic signal control," in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage. (CIKM), Beijing, China, Nov. 2019, pp. 1913–1922. doi:10.1145/3357384.3357902.

[3] E. van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in Proc. NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems, Barcelona, Spain, Dec. 2016.

[4] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO—Simulation of Urban Mobility," Int. J. Adv. Syst. Meas., vol. 5, no. 3–4, pp. 128–138, 2012.

[5] OpenStreetMap, [Online]. Available: https://www.openstreetmap.org [Accessed: May 15, 2025].

[6] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 4765–4774.

[7] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press

★★★