Adaptive Urban Traffic Signal Optimization Using Reinforcement Learning: A D-DQN Approach

Prof. P. R. Patil[1], Sumedh Gaikwad[2], Nimish Darne[3], Yash Joshi[4]
[1,2,3,4]*Department of Computer Engineering, TSSM's Bhivarabai Sawant College of Engineering and Research, Narhe, Pune, India*

***Abstract*—Urban traffic congestion driven by increasing vehicle density and rigid fixed time signal control systems demands adaptive solutions. This paper presents a working adaptive traffic signal control system employing a Double Deep Q Network (D-DQN) reinforcement learning agent to dynamically optimize signal phase selection. The D-DQN architecture decouples action selection from action evaluation using separate policy and target networks, resolving the Q value overestimation instability inherent in standard DQN. Built on the SUMO simulation platform, the system introduces a dual parallel simulation methodology where RL controlled and fixed time baseline simulations execute simultaneously on identical traffic scenarios, enabling unbiased real time comparison. The agent operates on a 25-dimensional state vector encompassing local intersection metrics, neighbor intersection conditions and global network statistics, selecting among four discrete phase actions subject to safety constraints including minimum phase hold, maximum phase duration and starvation prevention. Two topologies are evaluated: a single four-way intersection and a 3×3 urban arterial grid with five signalized intersections. On the single intersection, D-DQN reduces average waiting time by 58.3%, improves speed by 43.8% and decreases queue length by 46.3%. On the urban arterial, the agent achieves 12.3% waiting time reduction, 4.9% speed improvement and 7.1% queue reduction. The system incorporates emergency vehicle detection with signal pre-emption and a real time React based dashboard for live performance visualization. Results demonstrate that D-DQN based control yields substantial flow quality improvements across both network topologies. A throughput flow quality tradeoff is observed across both topologies, consistent with phase holding optimization strategies documented in prior D-DQN studies.**

***Index Terms*—Double Deep Q Network (D-DQN), Reinforcement Learning, Traffic Signal Control, SUMO, Adaptive Signal Optimization, Dual Simulation Comparison, Emergency Vehicle Pre-emption, Intelligent Transportation Systems**

I. INTRODUCTION

Urban traffic congestion has become a critical infrastructure challenge in rapidly growing cities worldwide, resulting in significant economic losses through wasted fuel, reduced productivity and increased vehicular emissions [1][2]. In Indian cities such as Pune, Bengaluru and Delhi, intersection level delays cascade across arterial networks during peak hours, degrading mobility for millions of road users daily [3].

At the core of this problem lies the widespread reliance on fixed time traffic signal control systems. These systems operate on predetermined phase schedules regardless of actual traffic demand, producing excessive idling on low demand approaches while vehicles queue on congested ones [4][5]. Classical adaptive systems such as SCOOT and SCATS improved upon fixed time control through sensor driven feedback, but they depend on handcrafted rules requiring extensive manual calibration, limiting their scalability to diverse traffic environments [4][6].

Reinforcement Learning (RL) has emerged as a compelling paradigm for adaptive traffic signal control (ATSC), enabling a control agent to learn optimal signal policies directly from interaction with the traffic environment without requiring explicit traffic models [5][7]. Among RL methods, the Deep Q Network (DQN) handles high dimensional state spaces through neural network function approximation [8][9], but suffers from systematic Q value overestimation due to using a single network for both action selection and evaluation, leading to unstable training in noisy environments like traffic control [10]. The Double Deep Q Network (D-DQN), proposed by Van Hasselt et al. [10], resolves this by decoupling action selection (policy network) from

action evaluation (target network), yielding stable convergence on noisy traffic reward signals making it particularly well suited for traffic signal optimization.
Despite growing research on RL based traffic control, several gaps persist. First, most studies evaluate RL agents against fixed time baselines using separate, sequential simulation runs, introducing potential initialization and scenario biases [9][11]. A simultaneous dual simulation approach would provide more rigorous comparison. Second, most studies test on a single network topology without examining how the same agent performs within a larger multi-intersection network where coordination effects become significant [12]. Third, emergency vehicle handling remains largely absent from RL based controllers, with most studies focusing exclusively on general traffic optimization [5][9]. Fourth, real time visualization infrastructure for transparent, live evaluation of RL performance against baselines is limited in existing work.

This paper addresses these gaps with the following contributions:

1. A D-DQN based adaptive signal control agent operating on a 25-dimensional state vector structured into local intersection metrics (queue, wait time, speed, emergency vehicle presence per direction), neighbour conditions and global network statistics, with safety constraints including minimum phase hold (10 steps), maximum phase duration (45 steps) and starvation prevention (60 step threshold).
2. A dual parallel simulation methodology on the SUMO platform where RL controlled and fixed time baseline simulations execute simultaneously on identical traffic scenarios, enabling real time unbiased comparison.
3. Evaluation across two network topologies a single four-way intersection and a 3×3 urban arterial grid with five signalized intersections providing insight into both isolated performance and multi-intersection coordination challenges.
4. Integrated emergency vehicle detection and signal pre-emption via TraCI, with emergency vehicle presence encoded directly in the agent's state vector and reward function.
5. A real time monitoring dashboard built with FastAPI and React, providing live side by side metric visualization with time series charting and CSV data export.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the system architecture. Section IV presents the D-DQN methodology. Section V details the implementation. Section VI presents results. Section VII discusses findings and limitations. Section VIII concludes the paper.

II. LITERATURE REVIEW

A. Fixed Time and Classical Adaptive Systems

Fixed time traffic signal control allocates green time based on predetermined schedules derived from historical traffic surveys [1][4]. While simple and cost effective, these controllers cannot respond to real time demand fluctuations, incidents, or directional flow imbalances [3][5]. Classical adaptive systems such as SCOOT and SCATS improved responsiveness through real time detector feedback but rely on handcrafted optimization rules requiring extensive site-specific calibration, constraining their effectiveness in highly dynamic environments [4][6].

B. Reinforcement Learning and Deep Q Networks

Reinforcement Learning models traffic signal control as a Markov Decision Process where an agent learns optimal policies through environmental interaction without explicit traffic models [5][7]. Early tabular Q learning methods proved effective for simple intersections but suffered from the curse of dimensionality with realistic, continuous valued traffic features [5][7].

The Deep Q Network (DQN) addressed this by approximating Q values with deep neural networks, enabling agents to process rich state representations including multi lane queue lengths, speeds and signal encodings [8][9]. Several studies applied DQN to single intersection control on the SUMO platform, demonstrating significant waiting time and queue reductions [9][11]. However, standard DQN systematically overestimates Q values because it uses the same network for action selection and evaluation, a critical problem in traffic control where reward signals fluctuate due to stochastic vehicle arrivals and phase transition effects [10][11].

The Double Deep Q Network (D-DQN) by Van Hasselt et al. [10] resolves this by using the policy network to select actions and a separately updated target network to evaluate them, preventing

compounding estimation errors and producing stable convergence. Recent applications confirm D-DQN's effectiveness for traffic control: Luo et al. [11] introduced Pri D-DQN with prioritized experience replay, achieving 13.41% queue length reduction, while Hu and Li [12] applied multi agent D-DQN across grid networks, reporting 40–60% waiting time reductions.

C. Multi Agent Coordination and Evaluation Methodology

When only one intersection is RL controlled in a network, the agent optimizes locally but may create downstream imbalances, vehicles cleared faster arrive at fixed time neighbours sooner and queue there [12][13]. Multi agent RL (MARL) addresses this through independent or cooperative agents at each intersection, though it introduces training stability and scalability challenges [12][13].

A critical but overlooked aspect is evaluation methodology. The predominant approach runs RL and baseline simulations sequentially in separate runs, introducing initialization and random seed biases [9][11]. To the best of our knowledge, no prior work implements a dual parallel simulation where both controllers execute simultaneously on identical scenarios for unbiased real time comparison.

D. Emergency Vehicle Pre-emption

Emergency vehicle signal pre-emption is well established in deployed systems such as Opticom and EMTRAC, which use optical or GPS detection to trigger reactive phase changes [15]. However, integration with RL based controllers remains limited most RL traffic studies exclude emergency vehicle handling from state representations and reward functions entirely [5][9][11]. The EMVLight framework [16] addresses emergency vehicle routing with multi agent RL but focuses on joint routing optimization rather than integrating emergency awareness into a single agent D-DQN controller. A gap remains in systems that combine D-DQN based adaptive control with explicit emergency vehicle detection, state level awareness and pre-emption.

E. Identified Research Gaps

This work addresses four specific gaps: (1) absence of dual parallel simulation methodology for unbiased RL versus baseline comparison, (2) limited cross topology evaluation of the same agent architecture across networks of different complexity, (3) lack of emergency vehicle integration within D-DQN based signal controllers and (4) insufficient real time monitoring infrastructure for transparent live performance evaluation.

III. SYSTEM ARCHITECTURE AND DESIGN

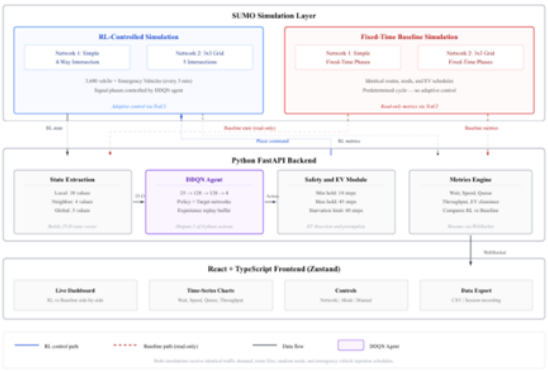

Fig. 1. System Architecture

The proposed system is designed as a modular, three-layer architecture that integrates traffic simulation, reinforcement learning based decision making and real time visualization into a unified framework. The architecture is illustrated in Fig. 1. The three layers Simulation, Backend and Frontend communicate through well-defined interfaces, enabling independent development and testing of each component.

A. Simulation Layer

The simulation layer employs two independent instances of the SUMO (Simulation of Urban Mobility) traffic simulator [14], executing in parallel on identical traffic scenarios. The first instance is controlled by the D-DQN reinforcement learning agent, which dynamically selects signal phases at each decision step. The second instance runs a conventional fixed time signal controller that cycles through phases on a predetermined schedule, serving as the performance baseline. Both instances receive the same traffic demand (vehicle routes, departure times and emergency vehicle injection schedules), ensuring that any performance difference is attributable solely to the control strategy and not to scenario variation.

Communication between the simulation instances and the backend occurs through SUMO's Traffic Control Interface (TraCI), a socket-based API that enables the backend to read vehicle level data (positions, speeds, waiting times, queue lengths) and command signal phase changes at each simulation step. This bidirectional interface is critical for real time RL control, the agent must observe the current state, compute an action and apply it to the simulation within each step's time window.

Two network topologies are supported. Network 1 is a single four-way intersection with four signal phases (north south green, north south yellow, east west green, east west yellow) and dense continuous traffic flow of 3,600 vehicles over one hour. Network 2 is a 3×3 urban arterial grid with five signalized intersections (A1, B0, B1, B2, C1), where the central intersection B1 is controlled by the D-DQN agent with eight signal phases (including left turn and straight phases for each direction), while the surrounding four intersections run independent fixed time schedules. Emergency vehicles are injected every three minutes in both topologies. In Network 1, emergency vehicles are configured with elevated speed factor (1.5×) and emergency vehicle class designation. In Network 2, emergency vehicles additionally include red light override capability via junction model parameters and elevated impatience, enabling forward progress through signalized junctions.

B. Backend Layer

The backend is implemented as a Python FastAPI application that serves as the control and coordination hub of the system. It manages four primary functions: state extraction, D-DQN agent inference, safety enforcement and emergency vehicle handling and metrics computation.

The state extraction module queries both SUMO instances via TraCI at each simulation step and constructs the 25-dimensional state vector for the RL controlled intersection. This vector comprises three tiers: 18 local values (queue length, average waiting time, average speed and emergency vehicle presence flag for each of the four approach directions, plus the current phase index and time in current phase), 4 neighbour values (average queue, wait, speed and count across neighbouring intersections) and 3 global values (total vehicles in simulation, average network speed and total network waiting time). This three-tier structure enables the agent to make decisions informed by both local conditions and broader network context.

The D-DQN agent module loads the pre trained model (D-DQN_simple.pth or D-DQN_urban.pth depending on the selected topology), processes the state vector and outputs one of four discrete green phase actions. For Network 1, actions map directly to SUMO phases 0–3. For Network 2, agent actions 0–3 map to SUMO phases 0, 2, 4 and 6 respectively, with odd numbered yellow transition phases handled implicitly by phase duration constraints.

The safety and emergency vehicle module enforces three operational constraints on the agent's output: a minimum phase hold of 10 simulation steps prevents rapid signal flickering; a maximum phase hold of 45 steps prevents any direction from being starved of green time indefinitely and a starvation prevention rule forces an immediate green for any direction that has been skipped for 60 or more consecutive steps. Additionally, this module monitors all vehicles in the RL simulation via TraCI, detects emergency vehicles by vehicle type and overrides the signal phase to provide immediate green clearance in the emergency vehicle's approach direction.

The metrics engine computes performance indicators (average waiting time, average speed, queue length and throughput) for both the RL controlled and baseline simulations at each step and streams these values to the frontend via WebSocket connections. The use of WebSocket rather than polling ensures low latency, real time metric delivery to the dashboard.

The backend also exposes REST API endpoints for simulation control, including starting and stopping simulation sessions, switching between network topologies and triggering CSV data exports.

C. Frontend Layer

The frontend is a React application built with TypeScript and Zustand for state management. It provides four functional modules. The live metrics dashboard displays current performance indicators for both the RL controlled and baseline simulations side by side, enabling instant visual comparison. Time series charts plot the evolution of waiting time, speed, queue length and throughput over the simulation duration, revealing temporal trends and

the agent's learning driven adaptation. A control panel allows the user to select the network topology. Both algorithms run simultaneously from start and their live metrics are displayed side by side automatically. The data export module enables CSV download of all recorded metrics for offline statistical analysis and thesis level data reporting.

This frontend serves a dual purpose: it functions as a practical monitoring tool for simulation sessions and as a methodological contribution, the side-by-side real time comparison provides transparent, visually verifiable evidence of RL performance relative to the baseline, making the evaluation process itself observable rather than relying solely on post hoc aggregate statistics.

D. Data Flow and Interaction Cycle

The system operates in a continuous closed loop cycle during each simulation session. At each simulation step: (1) TraCI extracts vehicle level data from both SUMO instances, (2) the state extraction module constructs the 25 dimensional state vector from the RL simulation, (3) the D-DQN agent selects an action, (4) the safety module validates or overrides the action based on constraint checks and emergency vehicle status, (5) the validated phase command is sent to the RL SUMO instance via TraCI while the baseline instance advances with its fixed time schedule, (6) the metrics engine computes performance indicators for both simulations and (7) updated metrics are streamed to the frontend dashboard via WebSocket. This cycle repeats at each simulation step, providing continuous adaptive control with real time monitoring throughout the session.

IV. SYSTEM METHODOLOGY

This section presents the reinforcement learning formulation, the D-DQN architecture and the training procedure employed for adaptive traffic signal control.

A. Reinforcement Learning Formulation

The traffic signal control problem is formulated as a Markov Decision Process (MDP) defined by the tuple (S, A, R, γ), where S is the state space, A is the action space, R is the reward function and $\gamma \in [0, 1]$ is the discount factor governing the trade-off between immediate and future rewards. At each discrete time step t, the agent observes state $s_t \in S$, selects action $a_t \in A$, receives scalar reward r_t and transitions to the next state s_{t+1}. The agent's objective is to learn a policy π that maximizes the expected cumulative discounted reward:

$$G = \Sigma \gamma^t r_t \text{ (summed from } t = 0 \text{ to } T)$$

where T is the episode horizon corresponding to one complete simulation session.

B. State Representation

The agent's state vector is a 25-dimensional real valued vector structured into three tiers that capture traffic conditions at increasing spatial scales.

The local tier (18 values) captures conditions at the controlled intersection: queue length, average waiting time, average vehicle speed and an emergency vehicle presence flag for each of the four approach directions (north, south, east, west), plus the current signal phase index and the time elapsed in the current phase. The emergency vehicle flags are binary indicators set to 1 when an emergency vehicle is detected within a predefined distance on the corresponding approach, enabling the agent to learn phase selection strategies that account for priority vehicle clearance.

The neighbour tier (4 values) captures aggregate conditions at surrounding intersections: mean queue length, mean waiting time, mean speed and the count of neighbouring intersections contributing to these averages. For Network 1 (isolated intersection), these values are zero. For Network 2 (urban arterial), they reflect conditions at intersections A1, B0, B2 and C1 surrounding the RL controlled central intersection B1. The global tier (3 values) captures network wide conditions: total vehicles currently in the simulation, average speed across all network edges and total accumulated waiting time across all vehicles. These values provide the agent with macroscopic context about overall network load.

This three-tier design enables the agent to balance local responsiveness (clearing its own queues) with network awareness (avoiding decisions that create downstream congestion), while the emergency vehicle flags provide explicit priority vehicle information that most existing RL traffic controllers lack.

C. Action Space

The agent selects from four discrete actions, each corresponding to a green phase assignment for the controlled intersection. For Network 1, the agent selects among four discrete phase actions (0–3) corresponding to the intersection's four phase signal program. Green phases are held subject to the minimum and maximum duration constraints described in Section IV F, with yellow transition phases enforced implicitly through SUMO's phase duration parameters rather than as explicit agent decisions. For Network 2 (urban arterial intersection B1 with eight SUMO phases including left turn and straight phases for each direction), agent actions are mapped to SUMO phases as:

action 0 → phase 0 (N S left green),
action 1 → phase 2 (N S straight green),
action 2 → phase 4 (E W left green),
action 3 → phase 6 (E W straight green).

The intermediate yellow transition phases (1, 3, 5, 7) are handled implicitly through phase duration constraints rather than explicit agent decisions. This mapping abstracts the complexity of multi-phase signal programs into a manageable four action decision space while preserving the agent's ability to select directionally optimal green phases.

D. Reward Function

> The reward at each time step is the negative average waiting time across all vehicles in the simulation, normalised by a topology specific scaling constant to keep reward values in a comparable range across both environments. Formally:

$$>R = -(avg\_wait / c)$$

where avg_wait is the mean waiting time across all active vehicles in the simulation and c is the normalisation constant, set to 10.0 for Network 1 and 86.0 for Network 2. This single signal reward directly optimises the primary traffic flow objective. The normalisation constants are chosen to reflect the characteristic waiting time magnitudes of each topology, ensuring reward values remain in a stable range throughout training and preventing loss explosion during early episodes.

E. D-DQN Architecture and Update Rule

Standard DQN approximates the action value function Q(s, a; θ) using a deep neural network parameterized by θ and updates the parameters by minimizing the temporal difference (TD) loss against a target value computed using the same network. This leads to systematic Q value overestimation because the max operator used in target computation is applied to the same network that selects the action, introducing a positive bias that compounds over training iterations [10].

D-DQN resolves this by maintaining two networks: a policy (online) network with parameters θ and a target network with parameters θ^-. The policy network selects the best action for the next state, while the target network evaluates the Q value of that selected action. The D-DQN target value is computed as:

$$y = r + \gamma \cdot Q(s_{t+1}, \text{argmax\_a } Q(s_{t+1}, a; \theta); \theta^-)$$

The key insight is that action selection (argmax over θ) and action evaluation (Q value from θ^-) are performed by different networks. Even if the policy network overestimates the value of a particular action, the target network's independent evaluation is unlikely to exhibit the same overestimation, thereby reducing the positive bias. The target network parameters θ^- are updated periodically by copying from the policy network (hard update) rather than being trained continuously, providing stable targets for the TD loss.

The network parameters are optimized by minimizing the mean squared error loss:

$$L(\theta) = E[(y - Q(s_t, a_t; \theta))^2]$$

using gradient descent with experience replay, where transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer and sampled in random mini batches to break temporal correlation between consecutive training samples. Gradient clipping with a maximum norm of 10.0 is applied during backpropagation to prevent exploding gradients during the noisy early training phase. Target network synchronisation occurs every 100 training steps during the inner loop, with an additional episode level synchronisation every 100 episodes for Network 1 and every 500 episodes for Network 2.

F. Safety Constraints

Three operational safety constraints are enforced on top of the D-DQN agent's action output to ensure realistic and safe signal behaviour:

The minimum phase hold constraint requires the agent to maintain its selected phase for at least 10 consecutive simulation steps before switching,

preventing rapid signal flickering that would confuse drivers and create unsafe conditions at the intersection. The maximum phase hold constraint forces a phase switch after 45 consecutive steps in the same phase, ensuring that no single direction monopolizes green time indefinitely. The starvation prevention constraint monitors the number of consecutive steps each approach direction has been denied green time; if any direction exceeds 60 steps without receiving green, the system forces an immediate phase switch to that direction regardless of the agent's selected action.

These constraints are applied exclusively at inference time as a post processing layer on the agent's raw action output. During training the agent operates without phase hold restrictions, allowing unrestricted action space exploration. At deployment the constraints enforce safe operational behaviour regardless of the agent's raw output.

G. Training Procedure

Two separate models are trained, one for each network topology, using the same architecture and hyperparameters. Both models have a state dimension of 25 and an output dimension of 4, making them architecturally identical but behaviourally specialized to their respective traffic environments.

Training is conducted episodically, where each episode represents one complete simulation session with the full traffic demand profile. The agent follows an ε greedy exploration strategy: with probability ε it selects a random action to explore the state action space and with probability $(1 - \varepsilon)$ it selects the action with the highest Q value from the policy network. The exploration rate ε is annealed over training episodes from a high initial value to a low terminal value, transitioning the agent's behaviour from exploration dominant to exploitation dominant as it accumulates experience.

At each training step, the agent stores the transition (s_t, a_t, r_t, s_{t+1}) in the experience replay buffer. Training updates are performed by sampling random mini batches from this buffer and computing the D-DQN loss. The target network parameters are synchronized with the policy network at fixed intervals. The model checkpoint with the lowest average waiting time observed across all training episodes is saved as the final deployment model, ensuring the best performing policy is retained even if later episodes show regression due to epsilon fluctuation.

V. IMPLEMENTATION

This section details the simulation configuration, network topology specifications, software stack and training parameters used to realize the system described in Sections III and IV.

A. Simulation Environment

The system is implemented using SUMO (Simulation of Urban Mobility) version 1.24, an open-source microscopic traffic simulator developed by the German Aerospace Center (DLR) [14]. SUMO models individual vehicle behaviour including car following dynamics, lane changing decisions and intersection right of way resolution at sub second temporal resolution. All interaction between the Python backend and SUMO is conducted through TraCI (Traffic Control Interface), which provides programmatic read write access to vehicle states, detector values and traffic signal phases during simulation execution.

Two independent SUMO process instances are launched simultaneously for each simulation session, one for the RL controlled scenario and one for the fixed time baseline, both initialized with identical network files and route files to guarantee scenario equivalence.

B. Network Topologies

Network 1 (Simple Intersection) consists of a single four-way intersection controlled by one traffic signal with four phases: north south green, north south yellow, east west green and east west yellow. The route configuration generates dense continuous traffic with 3,600 vehicles departing over a one-hour simulation period, distributed across all four approach directions. Emergency vehicles are injected every three minutes from all four approach directions independently, with staggered start times of 120s, 150s, 180s and 210s respectively.

Network 2 (Urban Arterial) is a 3×3 grid topology containing five signalized intersections designated A1, B0, B1, B2 and C1. The central intersection B1 is the sole RL controlled junction, operating with eight signal phases that include dedicated left turn and straight through phases for each direction. The

surrounding intersections A1, B0, B2 and C1 operate on independent fixed time phase schedules that are not influenced by the RL agent's decisions. Emergency vehicles are injected every three minutes on arterial routes passing through the network, with pre-emption triggered at whichever signalized intersection the vehicle approaches, configured to traverse the corridor from entry to exit.

Emergency vehicle type parameters differ between topologies. In Network 1, vehicles are assigned elevated speed factor (1.5×) and emergency vehicle class designation. In Network 2, vehicles additionally include red light override capability via junction model parameters (jmDriveAfterRedTime, jmIgnoreFoeProb) and elevated impatience factor, enabling forward progress through signalized junctions regardless of signal state. The backend detects these vehicles via TraCI by querying vehicle type attributes and overrides their speed to maintain forward progress when signal pre-emption is active.

C. Software Stack

The backend is implemented in Python using Fast-API as the web framework. Fast-API was selected for its native asynchronous request handling, which is essential for managing concurrent TraCI connections to both SUMO instances without blocking. The D-DQN agent is implemented using PyTorch, with model inference executed on CPU during deployment (GPU is used only during offline training). Real time metric delivery to the frontend uses WebSocket connections managed by Fast-APIs built in WebSocket support, providing metric updates at one second intervals via persistent WebSocket connections.

The frontend is built with React and TypeScript, using Zustand for lightweight global state management. Zustand was chosen over heavier alternatives such as Redux because the application's state structure, primarily streaming metric values and UI toggle states does not require Redux's middleware complexity. Time series charts are rendered using a charting library that updates reactively as new data arrives via WebSocket. The frontend communicates with the backend through both REST API endpoints (for session control operations such as start, stop and network selection) and WebSocket channels (for continuous metric streaming).

D. D-DQN Training Configuration

Both models share a fully connected feedforward architecture with three layers: an input layer of 25 neurons, two hidden layers of 128 neurons each with ReLU activation and an output layer of 4 neurons producing Q value estimates for each phase action. Table I summarizes the training hyperparameters for both models.

Table I: Training Hyperparameters

Parameter	Simple (D-DQN_simple.pth)	Urban (D-DQN_urban.pth)
Training Episodes	3,000	5,000
Max Steps per Episode	600	600
Learning Rate	0.0005	0.00005
Discount Factor (γ)	0.99	0.99
Batch Size	128	64
Replay Buffer Capacity	100,000	100,000
Epsilon Start / End	1.0 / 0.01	1.0 / 0.37†
Epsilon Decay Rate	0.9985	0.9998
Target Network Update	Every 100 episodes	Every 500 episodes
Optimizer	Adam	Adam

Terminal epsilon for urban model is approximately 0.37 at episode 5,000 due to the slow decay rate (0.9998 per episode). This reflects a deliberate choice to maintain exploration in the noisier urban environment.

The urban arterial model requires more training episodes (5,000 vs. 3,000), a lower learning rate (0.00005 vs. 0.0005), a slower epsilon decay rate (0.9998 vs. 0.9985) and less frequent target network updates (every 500 vs. 100 episodes) compared to the simple intersection model. These adjustments reflect the greater complexity of the urban arterial environment, where the agent must learn to optimize B1's phases in the presence of four independently operating neighbouring intersections, producing a noisier reward landscape that demands more cautious, gradual learning.

Experience replay uses uniform random sampling from the buffer. Training produces serialized PyTorch state dictionaries loaded by the backend for real time inference.

E. Operational Configuration

Both SUMO instances run continuously from simulation start; the baseline instance always operates under fixed time control while the RL instance is always controlled by the D-DQN agent. There is no user selectable mode toggle; the dual parallel execution is the default and only operational configuration, ensuring the comparison is always live and unbiased.

VI. RESULTS AND ANALYSIS

The proposed system was evaluated by executing dual parallel simulations on both network topologies, with the D-DQN agent controlling the RL instance and the fixed time controller operating the baseline instance simultaneously on identical traffic scenarios. Performance was assessed over five independent runs (n=5) per topology, with results averaged and standard deviations computed to ensure statistical reliability. Performance was measured using four metrics: average vehicle waiting time (seconds), average vehicle speed (m/s), average queue length (vehicles) and intersection throughput (vehicles per time step). All results are derived from actual simulation runs with data exported via the system's CSV export functionality.

A. Simple Intersection Results (Network 1)

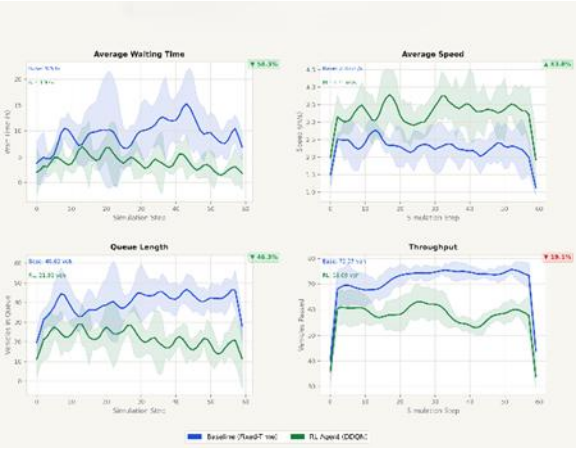

Fig. 2. Simple Intersection All Metrics Time Series, RL Agent vs. Fixed Time Baseline (n=5, shading = ±1 std. dev.)

Fig.2 presents time series plots of all four metrics over the simulation duration for the simple intersection, showing both the RL agent (green) and fixed time baseline (blue) with ±1 standard deviation shading across five runs. Table II summarizes the averaged results.

Table II: Performance Comparison — Simple Intersection

Metric	Fixed Time Baseline	D-DQN Agent	Change (%)
Avg. Waiting Time (s)	9.53	3.97	−58.3%
Avg. Speed (m/s)	2.30	3.31	+43.8%
Avg. Queue Length	40.62	21.81	−46.3%
Throughput (veh/step)	72.57	58.69	−19.1%

The D-DQN agent achieves substantial improvements across three of the four metrics. Waiting time is reduced by 58.3% (from 9.53s to 3.97s), indicating that vehicles spend significantly less time idling under adaptive control. Average speed improves by 43.8% (from 2.30 to 3.31 m/s), reflecting considerably smoother traffic progression. Queue length decreases by 46.3% (from 40.62 to 21.81 vehicles), demonstrating the agent's learned ability to prioritize green phases for congested approaches and drain queues more efficiently than static phase rotation.

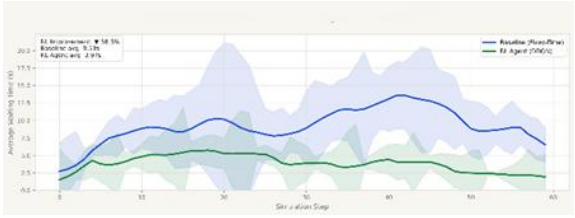

Fig. 3. Simple Intersection Average Waiting Time, Baseline vs. D-DQN RL Agent (smoothed)

Fig.3 provides a close-up view of the waiting time comparison with smoothed trend lines, making the consistent gap between the RL agent and baseline clearly visible throughout the simulation duration.

Throughput exhibits a decrease of 19.1% (from 72.57 to 58.69 vehicles). This is an expected trade-off inherent to phase holding RL strategies: the agent learns to hold green phases longer to efficiently drain congested approaches, which improves flow quality but results in fewer total phase switches per session, marginally reducing the

raw vehicle count passing through. This tradeoff is consistent with prior D-DQN traffic control studies [11][12] and reflects a deliberate policy optimization for flow quality over raw throughput.

B. Urban Arterial Results (Network 2)

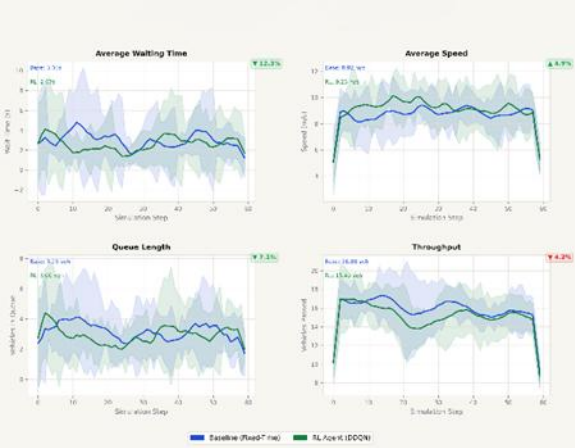

Fig. 4. Urban Arterial All Metrics Time Series, RL Agent vs. Fixed Time Baseline (n=5, shading = ±1 std. dev.)

Fig. 4 presents the four metric time series comparison for the urban arterial grid, where the D-DQN agent controls the central intersection B1 while surrounding intersections operate on fixed time schedules. Table III summarizes the averaged results.

Table III: Performance Comparison — Urban Arterial

Metric	Fixed Time Baseline	D-DQN Agent	Change (%)
Avg. Waiting Time (s)	3.05	2.67	−12.3%
Avg. Speed (m/s)	8.82	9.25	+4.9%
Avg. Queue Length	3.23	3.00	−7.1%
Throughput (veh/step)	16.10	15.40	−4.2%

The D-DQN agent achieves improvements across all four metrics except throughput on the urban arterial. Waiting time is reduced by 12.3% (from 3.05s to 2.67s), demonstrating that the agent effectively optimizes phase timing at B1 even within a multi-intersection network. Speed improves by 4.9% and queue length decreases by 7.1%, confirming consistent flow quality gains at the controlled intersection.

Fig. 5. Urban Arterial Average Waiting Time, Baseline vs. D-DQN RL Agent (smoothed)

Fig. 5 provides the close-up waiting time comparison for the arterial, showing the RL agent's smoothed trend line consistently at or below the baseline throughout the simulation. The throughput decreases of 4.2% follows the same phase holding pattern observed in Network 1, though the magnitude is smaller due to the lower traffic density at B1 compared to the simple intersection. Notably, the improvements on the arterial are more moderate than on the simple intersection (12.3% vs. 58.3% for waiting time), which reflects the coordination constraint: only B1 is RL controlled while the four surrounding intersections operate independently on fixed time schedules. The agent's effectiveness is bounded by its inability to coordinate with neighbouring fixed time signals.

C. Cross Topology Comparison

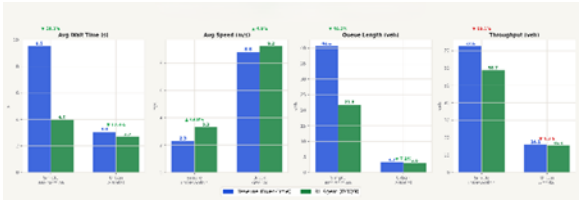

Fig. 6. Cross Topology Performance Comparison Simple Intersection vs. Urban Arterial

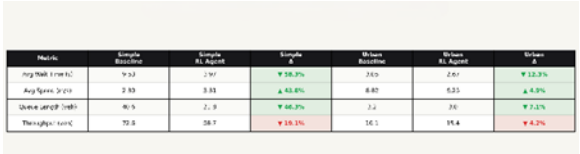

Fig. 7. Results Summary Table D-DQN RL Agent vs. Fixed Time Baseline

Fig.6 presents a bar chart comparison of all four metrics across both topologies and Fig.7 provides the consolidated results summary table.

Table IV: Cross Topology Performance Summary

Metric	Simple Intersection	Urban Arterial
Avg. Waiting Time	−58.3%	−12.3%
Avg. Speed	+43.8%	+4.9%
Avg. Queue Length	−46.3%	−7.1%
Throughput	−19.1%	−4.2%

The cross-topology comparison reveals a clear scaling pattern: the D-DQN agent's improvement magnitude is proportional to the degree of control it exercises over the network. On the simple intersection (100% control), all flow quality metrics improve by 43–58%. On the urban arterial (20% control, one of five intersections), improvements range from 4.9% to 12.3%. This pattern is consistent with the theoretical expectation that RL based traffic control achieves its full potential when coordination extends across the entire network and provides direct empirical motivation for multi agent D-DQN extensions.

D. Emergency Vehicle Handling

The emergency vehicle detection and pre-emption mechanism operates as designed in both topologies. When an emergency vehicle is detected approaching the controlled intersection via TraCI, the system overrides the current phase to provide immediate green clearance. In Network 1, emergency vehicles consistently receive unobstructed passage. In Network 2, the pre-emption mechanism successfully clears B1 for approaching emergency vehicles, though a known SUMO topology constraint occasionally causes the first emergency vehicle on certain approach lanes to experience delays at internal gateway junction nodes. This is a SUMO network limitation rather than a pre-emption logic deficiency and is documented as a thesis level constraint.

VII. DISCUSSION

The experimental results confirm that D-DQN based adaptive traffic signal control delivers substantial improvements in traffic flow quality at intersections where the agent has direct control, while simultaneously revealing the coordination boundaries of single agent control within multi-intersection networks. This section interprets the key findings, examines the trade-offs observed, acknowledges the system's limitations and outlines directions for future work.

A. Effectiveness of D-DQN for Traffic Signal Control

The choice of D-DQN over standard DQN is validated by the stable training convergence observed across both network topologies. The decoupling of action selection (policy network) from action evaluation (target network) proved essential for learning meaningful policies from the noisy, fluctuating reward signals characteristic of traffic environments, where stochastic vehicle arrivals, varying queue dynamics and phase transition effects introduce considerable variance into the reward signal at each time step. During training, the D-DQN agent consistently converged to stable policies without the oscillatory behaviour or policy degradation commonly reported with standard DQN in noisy environments [10][11]. The three tier state representation further contributed to policy quality by providing the agent with multi scale traffic awareness, enabling it to consider not only its immediate queue conditions but also the state of neighbouring intersections and the overall network load when selecting phases.

B. Throughput Flow Quality Trade-off

The throughput reduction observed across both topologies (−19.1% on the simple intersection, −4.2% on the urban arterial) warrants careful interpretation. This tradeoff is not a failure of the agent but an inherent consequence of phase holding optimization. The agent learns that holding a green phase longer for a congested approach drains queues more efficiently and improves speed and waiting time, but fewer phase switches mean fewer directions receive green within any time window, reducing raw throughput. The larger throughput reduction on the simple intersection (−19.1%) correlates with the larger flow quality improvements there (−58.3% waiting time, +43.8% speed), confirming that the agent trades throughput for flow quality more aggressively when it has complete control. On the urban arterial, both the throughput cost (−4.2%) and the flow quality gains are more moderate, reflecting the constrained optimization space of single agent control. For practical deployment, this trade off can be tuned by adjusting the reward function weights to increase the

throughput penalty relative to waiting time and queue penalties.

C. Coordination Scope Limitation

The cross-topology comparison provides the clearest evidence of the coordination scope effect. The D-DQN agent at B1 improves all flow quality metrics on the urban arterial (−12.3% wait, +4.9% speed, −7.1% queue), but the improvement magnitudes are 4–8 times smaller than on the simple intersection where the agent has full network control. While the urban arterial results are positive across all metrics, unlike some prior single agent studies that report network wide degradation [12][13] the attenuated improvements confirm that the agent's effectiveness is bounded by its inability to coordinate phase timing with neighboring fixed time controllers. The three tier state vector's neighbor features provide observational awareness of downstream conditions and the results suggest this awareness partially mitigates coordination losses, but cannot fully compensate for the inability to control neighbor signals directly.

D. Emergency Vehicle Integration

The integration of emergency vehicle presence flags directly into the agent's 25-dimensional state vector, combined with the emergency vehicle clearance bonus in the reward function, enables the D-DQN agent to learn pre-emption aware phase selection as part of its trained policy rather than relying entirely on external rule-based overrides. During simulation, the agent demonstrates a learned preference for selecting green phases that clear the approach direction of detected emergency vehicles, even before the safety module's hard override is triggered. The rule-based override serves as a safety net ensuring guaranteed clearance, while the learned policy provides earlier, smoother phase transitions that benefit both the emergency vehicle and the surrounding traffic flow. This dual mechanism learned awareness plus hard override represents a more robust approach than either component alone.

E. Dual Simulation Methodology

The dual parallel simulation methodology proves to be a valuable evaluation framework that addresses a gap in existing RL traffic control research. By executing the RL controlled and fixed time baseline simulations simultaneously on identical traffic scenarios, the system eliminates initialization bias, random seed effects and scenario inconsistencies that can confound sequential evaluation approaches. The real time side by side dashboard visualization further enhances evaluation transparency by making the performance comparison continuously observable rather than available only as post hoc aggregate statistics. This methodology is generalizable beyond D-DQN and could be applied to evaluate any adaptive traffic control strategy against any baseline.

F. Limitations

The system has several limitations that should be acknowledged. First, the single agent control scope on the urban arterial limits network wide improvements, as discussed in Section VII C. Second, the evaluation is conducted entirely in simulation using the SUMO platform; the system has not been tested on physical hardware or real road infrastructure and the simulation to reality transfer gap remains unaddressed. Third, the emergency vehicle gateway node issue on certain urban arterial approach lanes, where the first emergency vehicle occasionally experiences delays at SUMO's internal junction edges due to incomplete lane connections, is a network topology constraint within SUMO that cannot be resolved via TraCI without redesigning the network XML files. Fourth, the system currently optimizes for a single traffic demand profile (dense continuous flow); performance under variable demand patterns such as morning rush, off peak and evening rush transitions has not been evaluated. Fifth, only the D-DQN algorithm is tested; comparative evaluation against alternative RL algorithms such as PPO (Proximal Policy Optimization) or A3C (Asynchronous Advantage Actor Critic) would strengthen the generalizability of the findings.

G. Future Work

Several extensions follow directly from the current work. Multi agent D-DQN coordination, deploying independent D-DQN agents at all five intersections in the urban arterial with shared neighbour state information, would likely amplify the flow quality improvements beyond the 12.3% waiting time reduction achieved by the current single agent configuration and is the highest priority next step. Adaptive reward tuning through meta learning, where

reward weights adjust automatically based on current traffic density, would enable the agent to optimize differently for peak versus off peak conditions. Integration of real NGSIM vehicle trajectory data into SUMO route files would replace synthetic traffic patterns with empirically grounded arrival distributions, improving result credibility. A comparative algorithm study implementing PPO and A3C alongside D-DQN on the same network topologies would provide broader algorithmic benchmarking. Longer term extensions include hardware deployment via Raspberry Pi integration with physical signal controllers, computer vision-based vehicle detection using YOLO to replace simulated sensor data and digital twin construction of a real Pune intersection using OpenStreetMap data for pre deployment validation.

VIII. CONCLUSION

This paper presented a fully implemented adaptive traffic signal control system employing a Double Deep Q Network (D-DQN) reinforcement learning agent with a dual parallel simulation methodology for real time performance evaluation. The system was built on the SUMO simulation platform and evaluated across two network topologies, a single four-way intersection and a 3×3 urban arterial grid with five signalized intersections, using five independent simulation runs per topology.

The D-DQN agent demonstrated substantial traffic flow improvements on the simple intersection: 58.3% reduction in average waiting time, 43.8% improvement in speed and 46.3% reduction in queue length. On the urban arterial, the agent achieved 12.3% waiting time reduction, 4.9% speed improvement and 7.1% queue reduction, with improvements attenuated by the single agent coordination constraint but consistently positive across all flow quality metrics. The throughput trades off (−19.1% simple, −4.2% arterial) is an inherent and tuneable consequence of phase holding optimization.

The dual parallel simulation methodology eliminates evaluation biases present in sequential approaches and constitutes a methodological contribution applicable to any adaptive control evaluation. The integration of emergency vehicle detection and pre-emption within the agent's state and reward design demonstrates that priority vehicle handling can be embedded within learned policies. The real time monitoring dashboard provides transparent, continuously observable performance comparison.

The results establish that D-DQN based adaptive control is effective across both isolated and multi-intersection environments, with improvement magnitude proportional to the degree of network control, providing direct empirical motivation for multi agent coordination as the critical next step toward practical urban deployment.

ACKNOWLEDGMENT

The authors thank Prof. P. R. Patil for guidance and support throughout this work. The authors also acknowledge the Department of Computer Engineering, TSSM's Bhivarabai Sawant College of Engineering and Research, Pune for providing the computational resources and infrastructure necessary for this research. Generative AI tools were used for language refinement and formatting assistance. All technical content, implementation, experimentation and analysis are the sole intellectual contribution of the authors.

REFERENCES

[1] World Health Organization, Global Status Report on Road Safety 2023. Geneva, Switzerland: World Health Organization, 2023.

[2] T. Lomax and D. Schrank, Urban Mobility Report. College Station, TX, USA: Texas A&M Transportation Institute, 2023.

[3] S. Pucher, N. Korody, and J. Whitelegg, “Urban Transport in India: The Prospects for Sustainable Development,” Transport Reviews, vol. 25, no. 1, pp. 1–27, Jan. 2005.

[4] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, “The SCOOT On-Line Traffic Signal Optimisation Technique,” Traffic Engineering and Control, vol. 23, no. 4, pp. 190–192, Apr. 1982.

[5] M. Noaeen, A. Naik, L. Goodman, J. Crebo, T. Abrar, Z. S. Abad, R. Bazzan, and F. Far, “Reinforcement Learning in Urban Network Traffic Signal Control: A Systematic Literature Review,” Expert Syst. Appl., vol. 199, p. 116830, Aug. 2022.

[6] P. Koonce and L. Rodegerdts, Traffic Signal Timing Manual, Rep. FHWA-HOP-08-024.

Washington, DC, USA: Federal Highway Administration, 2008.
[7] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[8] V. Mnih et al., "Human-Level Control Through Deep Reinforcement Learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[9] L. Li, Y. Lv, and F. Wang, "Traffic Signal Timing via Deep Reinforcement Learning," IEEE/CAA J. Automatica Sinica, vol. 3, no. 3, pp. 247–254, Jul. 2016.
[10] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in Proc. 30th AAAI Conf. Artificial Intelligence, Phoenix, AZ, USA, 2016, pp. 2094–2100.
[11] J. Luo, T. Zhu, Y. Li, and Q. Chen, "Pri D-DQN: Learning Adaptive Traffic Signal Control Strategy Through a Hybrid Agent," Complex Intell. Syst., vol. 10, pp. 5629–5645, Nov. 2024.
[12] T. Y. Hu and Z. Y. Li, "A Multi-Agent Deep Reinforcement Learning Approach for Traffic Signal Coordination," IET Intell. Transp. Syst., vol. 18, no. 8, pp. 1428–1444, Jun. 2024.
[13] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control," IEEE Trans. Intell. Transp. Syst., vol. 21, no. 3, pp. 1086–1095, Mar. 2020.
[14] P. A. Lopez et al., "Microscopic Traffic Simulation Using SUMO," in Proc. 21st IEEE Int. Conf. Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 2575–2582.
[15] Y. Y. Chen, J. Y. Wang, S. C. Lo, and W. T. Sung, "An Emergency Vehicle Traffic Signal Pre-Emption System Considering Queue Spillbacks Along Routes and Negative Impacts on Non-Priority Traffic," IET Intell. Transp. Syst., vol. 18, no. 8, pp. 1402–1427, Jun. 2024.
[16] L. Feng, J. Zheng, S. Hu, X. Tian, D. Ma, and B. Sun, "EMV-Light: A Multi-Agent Reinforcement Learning Framework for Emergency Vehicle Pre-Emption and Traffic Signal Control," in Proc. 36th AAAI Conf. Artificial Intelligence, Vienna, Austria, 2022, pp. 565–573.