Adaptive Traffic Signal Control with Deep Q-Learning

Izhar Ali [1] **Million Haileyesus** [1]

Abstract

Traditional fixed-timing methods fail to provide optimal solutions for traffic signal control—a complex sequential decision problem. A Deep Q-Network (DQN) agent is presented for adaptive traffic signal control, implemented and evaluated in the SUMO simulation environment. Our system formulates the control problem as a Markov Decision Process with a state space of binary lane presence indicators, actions corresponding to traffic signal phases, and a reward function designed to minimize vehicle wait times. Experimental results demonstrate the agent's ability to progressively refine control policies, achieving a 99.7% reduction in total waiting time and 95.8% reduction in queue length compared to fixed-timing control, while also outperforming standard actuated control by 64.5% in waiting time reduction.

1. Introduction

Urban traffic signal control presents a fundamental challenge in transportation systems. The inherent stochasticity of traffic flows and the sequential nature of control decisions make this problem suitable for a Markov Decision Process (MDP) formulation. We define this MDP as the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where $\mathcal{S}$ represents possible traffic states, $\mathcal{A}$ denotes available signal configurations, $P(s'|s, a)$ specifies transition probabilities, $R(s, a)$ quantifies immediate rewards, and $\gamma \in [0, 1)$ balances immediate versus future rewards.

Traditional control approaches rely on fixed timing patterns or simple actuated logic that react to current conditions without optimization for long-term performance. These approaches implement policies $\pi : \mathcal{S} \to \mathcal{A}$ that lack the ability to adapt to dynamic traffic patterns. In contrast, the objective of intelligent traffic control is to find an optimal policy π^* that maximizes expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \tag{1}$$

which represents the goal of finding signal timing patterns that minimize overall vehicle waiting times across varying traffic conditions.

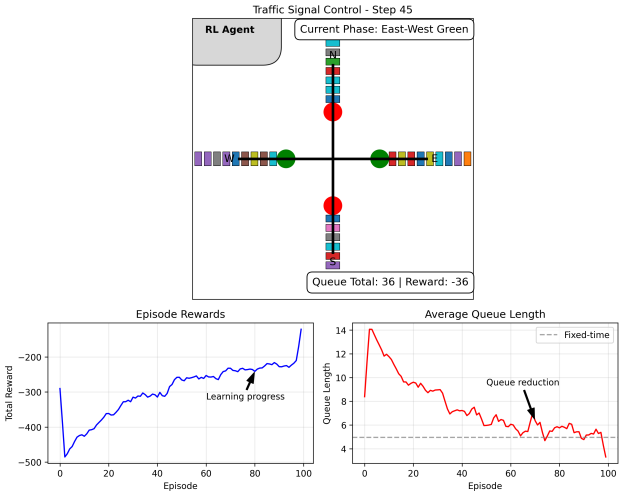

Figure 1. **Top:** simulated intersection with RL agent controlling signal phases (green east-west, red north-south). **Bottom:** Learning performance metrics showing increasing total reward (left) and reducing queue lengths (right) over training episodes compared to a fixed-timing baseline.

Three primary challenges complicate this optimization: stochastic traffic arrivals, high-dimensional state spaces, and delayed reward feedback. Reinforcement learning (RL) offers a principled approach to learn optimal policies through direct interaction with the environment without requiring explicit transition models. We employ a Deep Q-Network (DQN) that approximates the optimal action-value function $Q^*(s, a)$ using a neural network architecture. This function quantifies the expected future reward of each action in each state, enabling the agent to select optimal actions.

[1]Department of Computer Science, Rowan University, Glassboro, NJ, USA. Correspondence to: Izhar Ali <aliizh94@students.rowan.edu>, Million Haileyesus <hailey74@rowan.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Building on work by (Li et al., 2016), we implement a DQN agent using TensorFlow and Keras to optimize traffic flow at a single intersection within the SUMO (Simulation of Urban MObility) simulation environment. The agent minimizes aggregate vehicle waiting times by learning from interactions with the simulated traffic system. Experience replay enables learning from diverse scenarios, contributing to more robust and generalizable control policies.

2. Related Work

Research on reinforcement learning for traffic signal control can be categorized into three principal approaches: classical RL methods, deep RL applications, and multi-agent coordination systems.

Classical RL Approaches. Early work by (Thorpe & Anderson, 1997) applied SARSA to single-intersection control, establishing fundamental action-state designs. (Wiering, 2000) extended this to multiple intersections with coordinated Q-learning. Tabular representations limited these approaches to simple traffic scenarios.

Deep RL for Enhanced Representation. The integration of deep learning with RL transformed traffic control capabilities by enabling learning from high-dimensional states. (Li et al., 2016) first applied DQN to traffic control, adapting the architecture developed by (Mnih et al., 2015) and demonstrating quantifiable improvements over fixed-timing methods. Their contribution established effective state representations and reward functions specifically designed for traffic control. (Wei et al., 2018) advanced this approach with an actor-critic architecture that further improved performance through hierarchical action selection.

Multi-Agent Coordination. For network-scale traffic management, several researchers have developed multi-agent reinforcement learning (MARL) frameworks. (Chu et al., 2019) implemented information-sharing mechanisms between intersection agents to achieve network-level optimization. These approaches address the coordination problem that emerges when multiple RL controllers operate in interconnected traffic networks.

Conventional Baselines. Fixed-timing methods (Koonce et al., 2008) and actuated control systems (Zhao et al., 2011) remain important performance benchmarks. Systems like SCOOT (Hunt et al., 1981) represent sophisticated rule-based approaches but lack the adaptive learning capabilities of RL methods.

Our work provides an implementation of a DQN agent for single-intersection control, with specific focus on precise state representation, discrete action selection, and a waiting-time-based reward function. We evaluate this system in SUMO using procedurally generated traffic scenarios.

3. System Description

3.1. Problem Formulation

The traffic signal control problem is formulated as a Markov Decision Process (MDP) characterized by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with the following components:

- State $\mathcal{S}$: The state $s \in \mathcal{S}$ is a vector of 80 binary values, representing vehicle presence or absence in discretized cells along the approach lanes of a four-way intersection. Each of the four approach arms is divided into 20 cells: 10 cells for the dedicated left-turn lane and 10 cells for the combined through and right-turn lanes. This discretization allows the agent to perceive the spatial distribution of queuing vehicles. The cells are non-uniformly sized, with finer granularity (shorter cells) closer to the intersection to provide more precise information where control decisions are most critical, enhancing sensitivity to queue formation near the stop line.

- Action $\mathcal{A}$: The action $a \in \mathcal{A}$ is a discrete choice from a set of 4 predefined signal phases: (1) North-South (N-S) through/right green, (2) N-S left-turn green, (3) East-West (E-W) through/right green, and (4) E-W left-turn green. When an action is selected, the corresponding green phase is activated for a fixed duration $T_G = 10$ seconds, which is then followed by a yellow transition phase of duration $T_Y = 4$ seconds before the next state is observed and a new action can be taken.

- Transition Probability $P(s'|s, a)$: This function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, models the stochastic dynamics of traffic flow. It is implicitly defined by the SUMO microscopic traffic simulator, which governs how the traffic state s evolves to a new state s' after action a is executed.

- Reward Function $R(s, a)$: The reward R_t at decision step t is defined as the change in the cumulative waiting time of all vehicles on the incoming lanes:

$$R_t = W_{t-1} - W_t \tag{2}$$

where W_t is the sum of waiting times (seconds a vehicle has speed zero since spawn) of all vehicles on approach lanes at the current decision point t, and W_{t-1} is the corresponding sum from the previous decision point. A positive reward thus indicates a reduction in total waiting time, directly incentivizing the agent to improve traffic flow and reduce vehicle idling. This formulation provides immediate feedback for actions while addressing the credit assignment challenge inherent in traffic control.

- Discount Factor γ: A discount factor (default 0.75) that weights the importance of immediate rewards versus future rewards. A value less than 1 encourages the agent to find solutions that are beneficial in the long term.

3.2. System Architecture

The core of the architecture is a custom simulation environment interface that wraps SUMO's functionality. This interface manages the simulation lifecycle, extracts state information, translates agent actions into traffic signal commands, and computes rewards based on vehicle waiting times. Communication with SUMO occurs through the TraCI (Traffic Control Interface) library over a TCP socket.

To generate diverse traffic conditions, a traffic scenario generator creates SUMO-compatible route files that define vehicle types, routes, and arrival patterns. This module uses stochastic distributions to model realistic traffic flows, particularly a Weibull distribution for vehicle arrivals that mimics natural traffic density fluctuations.

The architecture enables the agent to perceive the environment through 80 binary state values representing vehicle presence in discretized road segments. Based on this representation, the agent selects one of four possible traffic signal phases, which are then executed within the SUMO environment.

3.3. Training Framework

Training begins with initialization of the DQN model parameters θ and an empty experience replay memory $\mathcal{D}$. Configuration parameters including total training episodes M_E and maximum simulation steps per episode T_{max} are loaded. The exploration rate ϵ is set for initial random exploration and gradually decreased as training progresses.

For each of the M_E episodes, the framework resets the simulation environment and generates a new traffic scenario. At each decision point within an episode, the agent observes the current state s_t and selects an action a_t using an ϵ-greedy policy, balancing exploration of new actions with exploitation of learned knowledge. If this action differs from the previous one, the system first activates a yellow transition phase for duration T_Y before implementing the new green phase for duration T_G. After executing the action, the agent observes the resulting reward r_t (based on changes in vehicle waiting times) and new state s_{t+1}. This transition tuple (s_t, a_t, r_t, s_{t+1}) is stored in the replay memory $\mathcal{D}$.

Network training occurs after each episode, provided the replay memory contains sufficient samples (N_{min}). Algorithm (1) performs N_{train} iterations, each sampling a minibatch of N_B transitions from $\mathcal{D}$. For each sampled transition, a target Q-value is computed using the Bellman equation, which bootstraps current estimates from future states:

$$y_k = r_k + \gamma \max_{a'} Q(s_{k+1}, a'; \theta) \tag{3}$$

The network parameters are then updated through gradient descent to minimize the mean squared error between predicted and target Q-values:

$$\mathcal{L}(\theta) = \frac{1}{N_B} \sum_k (y_k - Q(s_k, a_k; \theta))^2 \tag{4}$$

The episodic training allows the agent to encounter a variety of traffic conditions while the experience replay mechanism ensures efficient use of past experiences, leading to progressive improvement in the traffic control policy.

3.4. Simulation Environment

We utilize SUMO (Simulation of Urban MObility) (Lopez et al., 2018), an open-source, highly portable, and microscopic traffic simulation package. SUMO is chosen for its capability to model individual vehicle dynamics with realism (including acceleration, deceleration, and lane-changing behaviors), its support for various traffic control mechanisms, and its well-documented TraCI (Traffic Control Interface) API that enables real-time interaction and control from external scripts, such as our Python-based RL agent.

3.4.1. Network Configuration

The simulation environment consists of a single four-way intersection as shown in Figure 2.

- Layout: The intersection features four approach arms (North, South, East, West), each comprising four 750-meter long incoming lanes and four outgoing lanes.
- Lane Behavior: For each approach, the lane configuration is as follows: the left-most lane is dedicated to left-turning vehicles, the right-most lane accommodates both right-turning and straight-through vehicles, and the two middle lanes are for straight-through movements only.
- Traffic Signal Control: The RL agent selects one of four principal green phases. If the chosen phase differs from the one previously active, the control script explicitly directs SUMO to first implement the yellow transition phase corresponding to the old action, before the newly selected green phase is activated. The durations of the green phase (T_G) and the preceding yellow phase (T_Y) are key configurable parameters (e.g., 10s and 4s respectively).

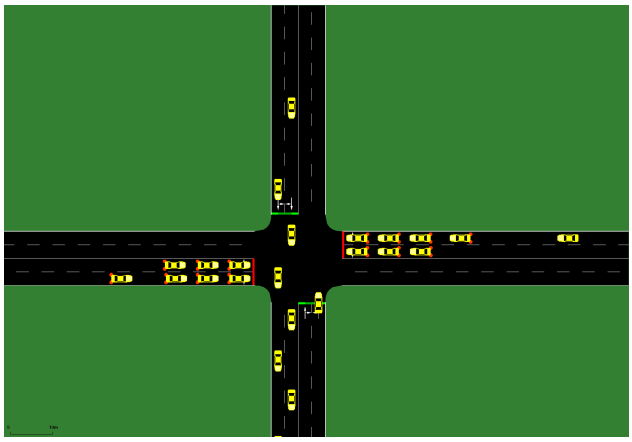

Figure 2. SUMO simulation visualization of the four-way intersection with vehicle queues forming on multiple approaches. Red highlights indicate significant queue formation on east-west approaches during a north-south green phase.

3.4.2. Traffic Generation

To provide diverse and realistic training conditions, traffic demand is programmatically generated for each simulation episode:

- Volume: A consistent number of vehicles (e.g., 1000 cars) is generated per episode.
- Arrival Pattern: Vehicle arrival times are modeled using a Weibull distribution with shape parameter $k = 2$. The Weibull distribution, with probability density function

$$f(x; \lambda, k) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} \quad x \geq 0 \qquad (5)$$

  produces a right-skewed distribution that effectively models the rise and fall of traffic intensity throughout a time period. This distribution is scaled to the total episode duration, resulting in an initial period of increasing traffic flow, followed by a period of decreasing flow, mimicking naturalistic variations in traffic density.
- Routing: Vehicle routes are assigned probabilistically: a significant majority (e.g., 75%) of vehicles are designated to travel straight through the intersection, while the remaining minority (e.g., 25%) are assigned to make turns (left or right, chosen randomly).
- Spawn Locations: Vehicles are introduced into the network with an equal probability from any of the four approach arms.
- Reproducibility and Variability: Traffic generation is randomized for each episode by using a unique seed (derived from the episode number). This ensures that the agent encounters a wide range of traffic scenarios during training, while also allowing for the exact replication of specific scenarios for testing or debugging purposes.

4. Deep Q-Network Algorithm

The core of our adaptive traffic signal control system is a Deep Q-Network (DQN) agent. DQN is a model-free, off-policy reinforcement learning algorithm that learns an optimal policy π^* by approximating the optimal action-value function $Q^*(s, a)$. The $Q^*(s, a)$ function represents the maximum expected discounted future reward achievable by taking action a in state s and following the optimal policy thereafter.

4.1. Implementation Details

Our DQN implementation includes:

- **Neural Network Architecture**: A fully connected network with an 80-unit input layer (matching the binary state representation), 5 hidden layers (400 units each with ReLU activation), and a 4-unit output layer with linear activation corresponding to the possible signal phases.
- **Learning Process**: The network minimizes temporal difference error using the Bellman equation

$$y = r + \gamma \max_{a'} Q_\theta(s', a')$$

  Our implementation uses a single network for both action selection and evaluation, simplifying the architecture while potentially allowing Q-value overestimation.
- **Training Mechanism**: We use experience replay to store transitions in a fixed-capacity buffer $\mathcal{D}$, randomly sampling minibatches during training to break temporal correlations. The Adam optimizer updates network parameters by minimizing Mean Squared Error loss between predicted and target Q-values.
- **Exploration Strategy**: An ϵ-greedy policy balances exploration and exploitation, with ϵ annealed linearly from 1.0 to a small value (e.g., 0.01) across training episodes, gradually transitioning from random exploration to learned policy exploitation.

5. Experimental Evaluation

In this section, we demonstrate how our MDP formulation, state representation, and learning algorithm perform in realistic traffic scenarios. Our experiments are designed to

Algorithm 1 Deep Q-Network (DQN) Training Algorithm

Initialize replay memory $\mathcal{D}$ to capacity N_M.
Initialize action-value function Q with random weights θ.
Set total episodes M_E. For 1-indexed episode i from 1 to M_E, calculate exploration rate $\epsilon_i = 1.0 - ((i-1)/M_E)$.
for episode $i = 1$ to M_E **do**
  Generate initial state s_1 for the current episode by resetting the simulation environment.
  $a_{prev} \leftarrow$ null (initialize previous action)
  for $t = 1$ to T_{max} (maximum steps per episode) **do**
    if a uniformly drawn random number from $[0, 1)$ is less than ϵ_i **then**
      Select a random action $a_t \in \mathcal{A}$ (Exploration).
    else
      Select $a_t = \arg\max_a Q(s_t, a; \theta)$ (Exploitation).
    end if
    Execute action a_t in the simulator:
    if $a_t \neq a_{prev}$ and $a_{prev} \neq$ null **then**
      Apply yellow signal phase for a_{prev} for duration T_Y.
    end if
    Apply green signal phase for a_t for duration T_G.
    Observe immediate reward r_t and next state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay memory $\mathcal{D}$.
    $s_t \leftarrow s_{t+1}$ (update current state).
    $a_{prev} \leftarrow a_t$ (update previous action).
  end for
  if current size of $\mathcal{D} \geq N_{min}$ (minimum memory size for training) **then**
    for $j = 1$ to N_{train} (number of training iterations this episode) **do**
      Sample a random minibatch of N_B transitions (s_k, a_k, r_k, s_{k+1}) from $\mathcal{D}$.
      For each transition k in the minibatch, calculate the target Q-value using the Bellman equation:

$$y_k = r_k + \gamma \max_{a'} Q(s_{k+1}, a'; \theta) \tag{6}$$

      Perform a gradient descent step on the loss $\mathcal{L}(\theta) = \frac{1}{N_B} \sum_k (y_k - Q(s_k, a_k; \theta))^2$ with respect to the network parameters θ.
    end for
  end if
end for

validate the agent's ability to learn effective control policies that minimize waiting times and improve traffic flow efficiency.[1]

5.1. Performance Metrics

We measure the following performance metrics for each episode:

- **Total Reward per Episode**: This is a primary indicator of learning. An improving (less negative) trend indicates that the agent is successfully learning to achieve higher rewards, corresponding to better traffic management.
- **Cumulative Waiting Time**: This is the sum of waiting times for all vehicles processed by the intersection during an episode. The agent's objective is to minimize this value; a decreasing trend indicates improved efficiency in reducing vehicle delays.
- **Average Queue Length**: This is the average number of stationary vehicles observed on the approach lanes, averaged over all simulation steps within an episode. Lower average queue lengths suggest reduced congestion and smoother traffic flow.

5.2. Results and Analysis

The DQN agent demonstrated progressive performance improvement throughout the training process, learning an effective policy for managing traffic flow. The agent's learning

[1] Code available at https://github.com/ali-izhar/traffic-signal-rl

progress is visualized by the performance metrics recorded over 250 training episodes.

Figure 3 shows the learning curve for cumulative negative reward. The agent successfully learned to increase its reward (reduce the negative value), indicating improved traffic management over time. The reward improved substantially during the initial training phase (approximately the first 200 episodes) before starting to plateau, suggesting convergence towards an effective policy.

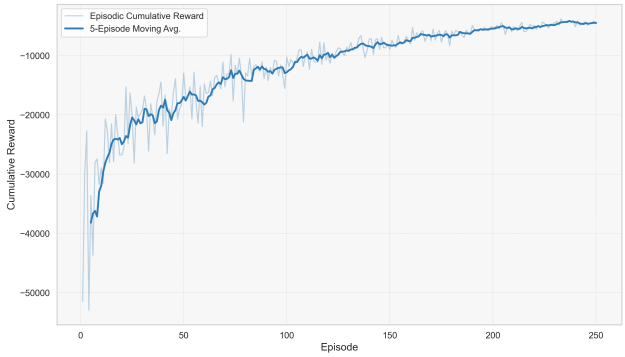

Figure 3. Learning progression shown through episode rewards.

Analysis of the cumulative delay metric (Figure 4) shows a corresponding significant reduction over the training episodes. This demonstrates the agent's ability to learn control strategies that effectively minimize vehicle waiting times by adapting signal timing to traffic conditions.

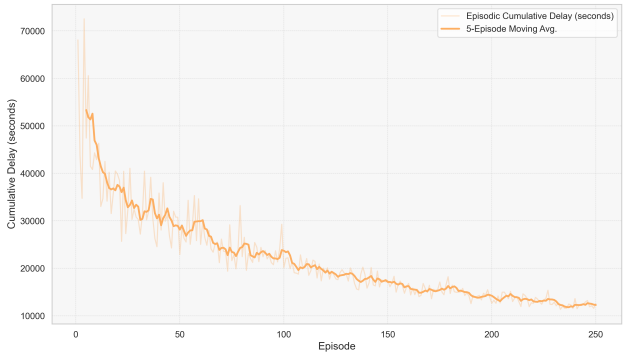

Figure 4. Reduction in vehicle delay over training.

Average queue length (Figure 5) decreased substantially during training, indicating reduced congestion at the intersection approaches. The agent learned to manage queues effectively, leading to smoother traffic flow. The stabilization of queue lengths in later episodes further supports the convergence of the learned policy.

The agent exhibited an evolving control strategy. In early

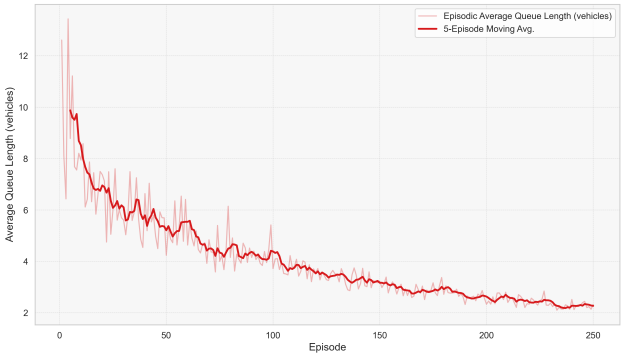

Figure 5. Queue management improvement during training: Average queue length drops from initial high values to consistently low levels by later episodes, demonstrating the agent's learned ability to prevent congestion buildup through anticipatory control.

episodes, it frequently changed signal phases, producing short green periods that failed to clear accumulated queues. As training progressed, the agent learned to adapt phase durations to traffic volume, allocating longer green times to approaches with heavier traffic and demonstrating more stable and effective control patterns in later episodes.

These results demonstrate that the DQN agent successfully learned an effective traffic signal control policy from interactions with the simulated environment, achieving significant reductions in key congestion metrics like delay and queue length.

5.3. Comparative Analysis

We compared our trained DQN agent against two baseline traffic control methods: a standard fixed-timing cycle and SUMO's default actuated control logic (generated via netconvert). Table 1 summarizes the performance across key metrics collected during a representative test episode using the default Weibull traffic distribution pattern for fair comparison.

Table 1. Performance comparison of control strategies. Wait Time (s) represents total accumulated wait time and Queue Length (vehicle/step) represents the average queue length.

Metric	Fixed-Time	Actuated	DQN (ours)
Wait Time (s)	76,299,328	635,811	**225,485**
Queue Length	58.24	5.35	**2.45**

The DQN agent demonstrated substantially superior performance compared to both baselines in this test scenario. It achieved a remarkable 99.7% reduction in total accumulated waiting time compared to the fixed-timing controller and a significant 64.5% reduction compared to the actuated con-

troller. For queue management, the DQN agent maintained average queue lengths 95.8% lower than the fixed-timing baseline and 54.2% lower than the actuated baseline.

Figures 6 and 7 visually illustrate this comparison, showing the metric values per simulation step for each control strategy against the background traffic demand intensity. The DQN agent consistently maintains lower waiting times and queue lengths throughout the simulation compared to both fixed-time and actuated control, particularly during peak demand periods.

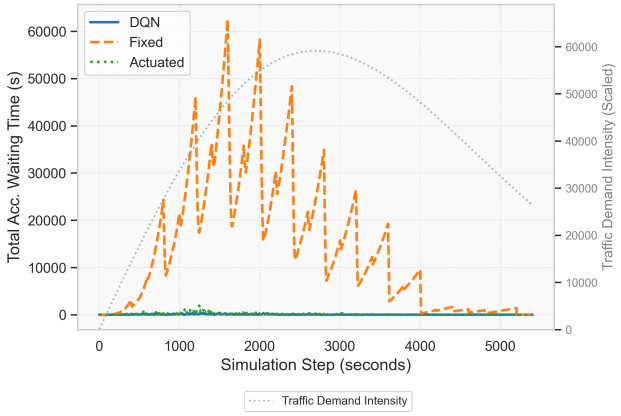

Figure 6. Total accumulated waiting time per simulation step.

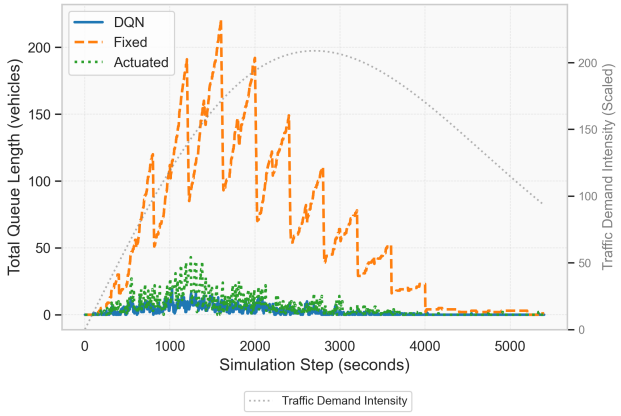

Figure 7. Total queue length per simulation step.

6. Discussion and Future Directions

Our DQN implementation for traffic signal control demonstrates the viability of deep reinforcement learning for adaptive intersection management. Three key findings emerge from our experiments:

First, the 80-cell binary state representation provides sufficient spatial granularity for effective control while remaining computationally tractable. This representation enables the agent to perceive detailed traffic distribution patterns across approach lanes, facilitating more nuanced signal timing decisions than would be possible with aggregate metrics alone.

Second, our waiting time differential reward function provides a direct learning signal that aligns with the practical goal of congestion reduction. The consistent improvement in performance metrics (99.7% reduction in waiting times compared to fixed-timing, 64.5% compared to actuated control) confirms the effectiveness of this formulation in guiding policy learning.

Third, the transition from random exploration to a stable, anticipatory control policy demonstrates successful convergence. The evolved policy exhibits sophisticated behaviors not explicitly programmed: phase duration adaptation based on traffic volume and predictive signal timing that anticipates developing traffic patterns.

Despite these successes, several limitations warrant acknowledgment:

- **Single-Intersection Scope:** Our implementation addresses an isolated intersection rather than a coordinated network, sidestepping the complex multi-agent coordination problems that arise in real-world traffic networks.
- **Simulation-Reality Gap:** While SUMO provides realistic traffic dynamics, real-world deployment would face additional challenges from sensor noise, unpredicted driver behaviors, and environmental factors not captured in simulation.
- **Algorithm Simplicity:** We implemented standard DQN rather than more advanced variants (Double DQN, Dueling architecture, or Prioritized Experience Replay) that might offer improved learning stability and performance. Additionally, other RL algorithms such as Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO) could potentially offer better performance through their continuous action spaces and more stable policy updates.

These limitations suggest two directions for future research:

First, extending to multi-intersection control using multi-agent reinforcement learning approaches would address real-world coordination needs. Techniques such as centralized training with decentralized execution or message-passing between agents could enable network-level traffic optimization.

Second, comparative evaluation against state-of-the-art traffic control systems (both conventional and RL-based) using

standardized traffic scenarios would provide more comprehensive performance benchmarking. This would clarify the specific conditions under which DQN-based control offers greatest advantages.

7. Conclusion

This paper presented a Deep Q-Learning approach to adaptive traffic signal control, implemented and evaluated within the SUMO traffic simulator. Our system combines a discrete lane-presence state representation, a waiting-time-based reward function, and standard DQN architecture with experience replay to learn effective control policies through direct interaction with simulated traffic.

Experimental results demonstrate substantial performance improvements over baseline methods. Compared to fixed-timing control, the trained DQN agent reduced total accumulated waiting time by 99.7% and average queue length by 95.8%. Notably, the DQN agent also significantly outperformed standard actuated control in the same scenario, achieving a 64.5% reduction in total waiting time and a 54.2% reduction in average queue length.

The performance demonstrated in this work suggests that deep reinforcement learning approaches can significantly outperform conventional traffic control methods, particularly in conditions of variable traffic demand where static or simple reactive strategies fail to adapt optimally. As transportation systems face increasing congestion challenges, adaptive learning-based controllers offer a promising path toward more efficient urban mobility.

References

Chu, T., Wang, J., Codecà, L., and Li, Z. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

Hunt, P., Robertson, D., Bretherton, R., and Royle, M. Scoot-a traffic responsive method of coordinating signals. *Transport and Road Research Laboratory*, 1981.

Koonce, P., Rodegerdts, L., Lee, K., Quayle, S., Beaird, S., Braud, C., Bonneson, J., Tarnoff, P., and Urbanik, T. Traffic signal timing manual. *Federal Highway Administration*, 2, 2008.

Li, L., Lv, Y., and Wang, F.-Y. Traffic signal optimization using reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 17(9):2474–2485, 2016.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. Microscopic traffic simulation using sumo. *IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2575–2582, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Thorpe, T. L. and Anderson, C. W. Vehicle traffic signal control using sarsa. *Proceedings of the IEEE International Conference on Neural Networks*, 4:2149–2154, 1997.

Wei, H., Zheng, G., Yao, H., and Li, Z. Intellilight: A reinforcement learning approach for intelligent traffic light control. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2496–2505, 2018.

Wiering, M. Multi-agent reinforcement learning for traffic light control. *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 1151–1158, 2000.

Zhao, J., Ma, W., Head, L., and Yang, X. Analysis of actuated traffic signal control systems. *Transportation Research Part C: Emerging Technologies*, 19(2):295–305, 2011.