2024 IEEE International Conference on Intelligent Signal Processing and Effective Communication Technologies (INSPECT) | 979-8-3503-7952-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/INSPECT63485.2024.10896157

Adaptive Traffic Signal Control System Using Deep Reinforcement Learning

Satyam Agrawal[1], Ritvij Sharma[1], Pankaj Srivastava[2], and Vinal Patel[3]

[1]Department of Information Technology
[2]Department of Engineering Sciences
[3]Department of Electrical and Electronics Engineering
Atal Bihari Vajpayee Indian Institute of Information Technology and Management, Gwalior
Email: {img_2021055, img_2021049, pankajs, vp}@iiitm.ac.in

***Abstract*—Traffic congestion remains a major issue in urban areas, leading to environmental deterioration through increased noise and air pollution. Existing traffic management solutions often fall short in adapting to the dynamic nature of urban traffic patterns. Adaptive traffic control systems utilizing Reinforcement Learning (RL) offer a promising alternative by adjusting traffic signals based on real-time traffic conditions in each lane to alleviate congestion. In this paper, we propose an RL-based approach leveraging Deep Q-Networks (DQN) to minimize the cumulative waiting time of vehicles at intersections. Our method involves training a deep neural network to predict Q-values for various traffic signal actions, which are then used to optimize traffic light changes. The simulation environment is set up using SUMO (Simulation of Urban Mobility) for network and traffic generation, with TraCI (Traffic Control Interface) providing real-time traffic data. Our approach demonstrates significant improvements over the traditional methods. Additionally, we include visualizations of the simulation results to illustrate the effectiveness of our RL-based traffic control system.**

***Index Terms*—Reinforcement Learning, Deep Q-Network, SUMO, ATSC**

I. Introduction

Traffic congestion has become a significant problem in cities around the globe, creating major challenges for commuters, urban planners, and policymakers. Rapid urban growth and increasing population densities have led to a rise in transportation demand, putting considerable strain on current traffic management systems. Conventional traffic control methods, which rely mainly on fixed-time signals, often cannot keep up with the changing and unpredictable nature of urban traffic. As a result, these methods lead to longer delays, higher fuel consumption, increased pollution, and greater accident risks.

Conventional traffic signal systems operate on preset timers, cycling through green, yellow, and red phases at fixed intervals, regardless of real-time traffic conditions [1]. This static approach may suffice in scenarios with consistent and predictable traffic patterns, but it falls short in addressing the complexities of modern urban traffic. During peak hours, fixed-time signals can lead to significant delays as they are not equipped to handle sudden surges in traffic volume. Conversely, during off-peak hours, vehicles may be forced to stop unnecessarily at red lights, wasting time and fuel. Such inefficiencies underline the need for a more adaptive and intelligent traffic management solution.

The advent of technology has paved the way for Adaptive Traffic Control Systems (ATCS), which leverage real-time data to optimize traffic signal timings. These systems utilize sensors embedded in road infrastructure to monitor traffic flow, vehicle counts, and other relevant parameters. Based on this data, ATCS dynamically adjusts the duration of green, yellow, and red lights to better match current traffic conditions. Cities like Bhubaneswar in India have already begun implementing such systems, leading to noticeable improvements in traffic flow and reductions in congestion. However, the implementation of ATCS often requires significant investment in infrastructure and maintenance, making it a challenging prospect for many cities.

To overcome these limitations and enhance the adaptability of traffic management systems, researchers are increasingly turning towards artificial intelligence, specifically Reinforcement Learning (RL). RL is a subset of machine learning where an agent learns to make decisions by interacting with its environment and receiving feedback in the form of rewards or penalties. In the context of traffic control, an RL agent can be trained to optimize traffic signal timings by continually learning from traffic patterns and outcomes.

An RL-based Adaptive Traffic Control System can operate by receiving real-time traffic data from sensors and using this information to make informed decisions about signal timings. The system aims to minimize overall traffic delays, reduce queue lengths, and enhance the flow of vehicles through intersections. Unlike fixed-time systems, an RL-based approach does not rely on predefined cycles but instead adapts continuously to changing traffic conditions.

One of the key advantages of using RL in traffic control is its ability to learn and improve over time. As the RL agent interacts with the traffic environment, it gathers valuable data on traffic patterns and the effects of different signal timings. This iterative learning process enables the system to develop

979-8-3503-7952-5/24/$31.00

more effective strategies for managing traffic, even in highly dynamic and unpredictable scenarios. The RL agent seeks to maximize a cumulative reward, which in this case could be defined as minimizing overall travel time, reducing stops, or balancing traffic loads across multiple intersections.

II. Related Work

Traditional traffic signal control methods, such as fixed-time control [2], SCOOT [3] , and SCATS [4], have been extensively implemented in real-world road networks, proving effective in managing intersections. However, their efficiency can diminish as traffic volume and complexity grow. To address the limitations of these traditional approaches, interdisciplinary methods such as fuzzy control [5] [6], genetic algorithms [7], and neural networks [8] [9] have been introduced into adaptive traffic signal control (ATSC) with promising outcomes. Nevertheless, each of these methods has its own challenges. For example, fuzzy control may face difficulties in handling complex multi-intersection situations, and genetic algorithms often require manual adjustment of parameters [10].

Current traffic signal control systems generally operate in one of three modes: predefined fixed schedules, adaptive/actuated control, or a combination of both. Predefined schedules rely on historical traffic data gathered through continuous monitoring with sensors like loop detectors. These sensors provide essential information, such as critical junctions and congestion patterns. However, because real-world traffic can be unpredictable, historical data alone may not capture all traffic irregularities. Alternatively, online traffic monitoring systems, such as those employing inductive loop sensors, can adjust signal timings in real-time based on current traffic conditions [11].

The Longest Queue First (LQF) scheduling algorithm is designed to minimize queue lengths at each approach to an intersection, aiming to decrease vehicle delays in comparison to existing signal control methods. This algorithm gives priority to specific vehicles, such as emergency responders or large trucks. Although its primary focus is on reducing queue lengths, ensuring the stability of the overall system remains a critical concern [12].

Traffic signal control can be conceptualized as a Markov Decision Process (MDP) [13]. In this context, Reinforcement Learning (RL) is applicable to Adaptive Traffic Signal Control (ATSC) because it does not require a predefined mathematical model of the external environment. Instead, RL algorithms interact with the environment and learn through experience, refining their parameters for optimal performance.Building on this foundation, researchers have developed a range of RL-based ATSC algorithms [14] .However, RL algorithms face limitations when dealing with complex and continuous state problems . Deep Reinforcement Learning (DRL) algorithms, which utilize deep neural networks, offer a solution to this challenge.

This paper advances the field by addressing a practical scenario: managing a single four-way intersection controlled by traffic lights using an autonomous agent. The agent observes current traffic conditions and applies insights gained from simulations to make informed decisions. The simulations utilize the Simulation of Urban Mobility (SUMO) tool, which provides a realistic yet controlled environment for testing various traffic regulation strategies. SUMO's Application Programming Interface (API) allows interaction with external programs, enabling real-time control of traffic lights based on the agent's decisions and the collection of traffic flow statistics. These statistics are used to establish a reward system for evaluating the agent's performance in managing traffic lights. We propose an RL-based approach employing Deep Q-Networks (DQN) to minimize the cumulative waiting time of vehicles at intersections. Our method involves training a deep neural network to predict Q-values for different traffic signal actions, which are then used to optimize traffic light changes.

III. Proposed Method

A. Background

Reinforcement learning, hereafter referred to as RL, has been proposed as a solution to this rising problem. There are two components of RL: the agent and the environment. The agent is the component that makes decisions, while everything else related to the problem is part of the environment. The environment mainly consists of states and rewards. In RL, agents are not told beforehand what actions to take. Instead, they learn the best actions by trying them and deciding based on the rewards they receive from the environment. Learning is based on trial and error. The agent starts from a fixed initial state, takes some actions, and receives rewards, which can be positive or negative, from the environment and then moves to the next state. The goal of the agent is to maximize the reward over the entire journey. The actions of the agents can have both immediate and future consequences.

RL methods are different from supervised learning algorithms. In supervised learning, the model learns from a training dataset. The dataset consists of the inputs as well as the final output. The model must learn to extrapolate its responses to be generalizable for a large number of inputs. One important concept in supervised learning is that there is no uncertainty in inputs or outputs. Supervised learning can be used to solve a large number of problems, but it fails in an interactive environment where the agent has no way of knowing the correct answer. The agent must be able to take actions in the face of uncertainty. On the contrary, the agent in RL is trying to maximize its reward after taking each action. The action of the agent may have consequences far into the future.

There are also four subcomponents of an RL system: policy, reward signal, and value function.

The policy is a mapping of states of the environment to the actions to be taken in those states. The policy can either be a lookup table or consist of search processes. The policy may also be stochastic, meaning it can assign probabilities to different actions.

The reward signal defines the goal of the RL problem and varies from problem to problem. It is the signal given to the agent whenever it executes an action. The reward is not usually known beforehand to the agent and can be positive or negative. The end goal of the agent is to maximize the reward signal it receives throughout the training process. In our paper, we define our reward signal as the difference between the cumulative delay time of the current episode and the cumulative delay of the last episode. The reward determines the change in policy. The agent performs its actions based on whether the reward gained was high or low.

In contrast to the reward signal, which indicates what action is good in the immediate future, the value function describes what is the right thing to do in the long run. Formally stated, the value of a state is the expected return an agent gains in the future, starting from the current state.

Deep Reinforcement Learning (DRL) enhances existing reinforcement learning (RL) models by integrating deep neural networks. These networks estimate Q-values for each state, identifying the optimal action to take to maximize the reward signal and overall return. Unlike traditional algorithms that rely on a Q-table, where a predefined function calculates Q-values for each state-action pair, DRL uses a neural network to approximate the best action given the current state.

Conventional RL approaches, like Q-learning, typically use value functions or policy-based strategies to determine optimal actions. However, these methods can struggle with high-dimensional state spaces or complex environments.

Deep reinforcement learning overcomes these challenges by employing deep neural networks to approximate value functions or policies. These networks, with their multiple layers and extensive parameters, enable DRL to capture complex relationships in the data, making it suitable for environments with intricate and continuous state and action spaces. For example, algorithms such as Deep Q-Networks (DQN) utilize deep neural networks to approximate Q-values, which predict the expected future rewards for taking specific actions in a given state.

Deep Q-Learning (DQN) represents an advanced technique in reinforcement learning that uses deep neural networks to estimate the Q-function, a critical element in value-based RL algorithms. In traditional Q-learning, the Q-function, Q(s, a), assesses the expected reward of performing an action, a, in a state, s, and then following the optimal policy thereafter. However, when dealing with large or continuous state spaces, maintaining a Q-table becomes impractical [15]. DQN addresses this issue by employing a neural network to approximate the Q-function, thereby adapting the problem for deep learning methodologies.

Simulation of Urban MObility, or SUMO for short, is an open-source, microscopic, multi-modal traffic simulation tool. It enables the simulation of individual vehicles as they navigate a specific road network based on a given traffic demand. SUMO supports a wide range of traffic management scenarios by explicitly modeling each vehicle, complete with its own route, and tracking its movement through the network [16].

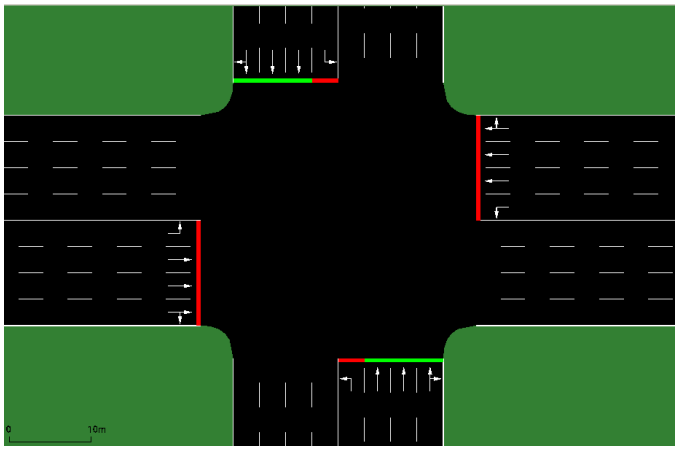


Fig. 1: 4-way intersection simulation environment in SUMO application

In this paper, we utilized SUMO to create our network and traffic demand models.

B. Simulation Settings

In this section of the paper we will describe the environment, states, model and its architecture, and the simulation settings.

Environment.As shown in Fig. 1, the environment is represented by 4 way intersections which contains 4 incoming as well as 4 outgoing lanes. The lanes are organized to accommodate specific traffic movements: the left-most lane on each arm is designated solely for left turns; the right-most lane allows for right turns and straight travel; while the two middle lanes are designated exclusively for straight-through traffic. The traffic light system at the intersection is organized such that the left-most lane on each arm has its own dedicated traffic light. In contrast, the remaining three lanes on each arm are controlled by a shared traffic light. This layout ensures that traffic flows efficiently while accommodating the various turning and straight-moving options available to drivers. In each episode, 1500 cars are generated with Weibull distribution. 75% percentage of cars will go straight, while remaining will turn either left or right with equal probability.

States. The state available to agent is given by the number of vehicles in each lane of the route. The state also consists of the speed of all the vehicles. The vehicles are considered stopped if they have speed less than 0.5 m/s.

Action. Agent has the choice to change the traffic light from 4 predetermined phases. Each phase a predetermined duration. Whenever the phase changes, yellow light is temporarily activated for 5 seconds. In any one step, the agent can allow the traffic to pass in the following directions: N-S, N-S Left, E-W and E-W Left.

Reward.

The agent's reward is based on the change in the cumulative waiting time of all vehicles across all lanes. Cumulative waiting time is calculated as the total waiting time of all vehicles that have not yet passed the intersection and are

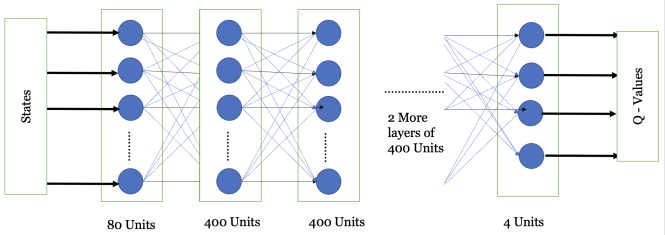


Fig. 2: Proposed DQN Model Architecture

stationary (speed of 0). Once a vehicle crosses the intersection, its waiting time is no longer included in the cumulative calculation. Thus, when a vehicle successfully crosses the intersection, it serves as a positive reward for the agent.

Q-Learning. Q-learning is a fundamental algorithm within reinforcement learning (RL) designed to tackle decision-making problems where an agent learns to select actions that maximize cumulative rewards through a process of trial and error. Unlike supervised learning, which relies on labeled data for training models, Q-learning enables agents to discover optimal policies by interacting directly with their environment. The key element of Q-learning is the Q-value function, Q(s,a), which estimates the expected future rewards for taking a particular action a in a given state s and then following the optimal policy thereafter. This function is updated using the Bellman equation, which connects the Q-value of the current state-action pair to the Q-values of the subsequent state. The update rule is expressed as:

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

where α is the learning rate, r is the received reward, γ is the discount factor, and $\max_{a'} Q(s', a')$ represents the maximum Q-value for the next state s'. A crucial aspect of Q-learning is balancing exploration and exploitation, typically managed through an ϵ-greedy policy, where the agent explores with probability ϵ and exploits the best-known action with probability 1-ϵ. Over time, as the agent interacts with the environment, it iteratively updates its Q-values based on its experiences, initially starting with arbitrary values.

C. Model

The proposed solution employs a Deep Q-Network (DQN) architecture. The model consists of an input layer, five hidden layers, and an output layer that provides the Q-values for each action. The hidden layers utilize the Rectified Linear Unit (ReLU) activation function, while the output layer uses a linear activation function. The input layer comprises 80 units, corresponding to the number of states. The processed input is then passed through the hidden layers, each containing 400 units. Finally, the output layer generates the 4 Q-values, each representing a possible action, as shown in Fig. 2.

D. Training

Firstly, we generate our network file in SUMO. It consists of 4-way intersections with 4 incoming and 4 outgoing lanes, with one lane dedicated to left turns. There are two traffic lights: one for 3 lanes and one for the dedicated lane. After that, traffic is generated with 1500 cars following a Weibull distribution. Each car has a 75% probability of going straight and a 25% probability of taking a left turn. The training process consists of 160 episodes, with each episode concluding when all vehicles have exited the simulation. Each traffic signal is programmed with a minimum green light duration of 10 seconds and a yellow light duration of 5 seconds to reduce the risk of collisions. Our model is a Deep Q-Network (DQN) that includes 1 input layer, 4 hidden layers, and 1 output layer. The input layer receives the current state, while the output layer generates the estimated Q-value for each possible action. The agent has 4 actions to choose from: allowing North-South traffic, permitting left turns for vehicles traveling North-South, and similarly for East-West traffic.

In the simulation process, the role of memory is crucial for managing and utilizing experiences from interactions with the environment. During each simulation run, the system collects data in the form of experiences, which include the state of the environment, the action taken, the reward received, and the resulting new state. This information is stored in memory, allowing the system to maintain a record of past interactions. Memory serves as a repository for these experiences, continuously updating as new interactions occur. The stored experiences are later used to improve the learning process of the reinforcement learning model. This method, known as experience replay, helps the model learn from a diverse set of past experiences rather than just the most recent ones. It stabilizes the learning process by reducing the correlation between consecutive experiences and provides a broader perspective on the environment.

A discount factor of 0.75 has been used to balance recent and past experiences. A decaying epsilon-greedy strategy has also been used, which depends on the number of episodes and decreases from 1 to 0 with a decrement of 1 / (number of episodes).

E. Simulation

At the start of the simulation, the model and the configuration settings are loaded. The initial reward is set to 0. TraCI is used to get the details of each car, including their speed, lane, and direction. The total waiting times for the vehicles are calculated. Current state, action, reward, and next state are then stored in memory. The agent executes an action using the epsilon-greedy strategy with the DQN model, using the current state as the input. The state is represented by an array of 80 elements based on the lane and the distance from the traffic light. The reward is then added to the cumulative reward store.

IV. Results and Discussion

This section is organized as follows: first, we will discuss the parameters used to evaluate our model;thereafter, we will compare our model to traditional traffic signal control systems. Additionally, we will also show how our model's performance is stable by comparing its performance with 3-way and 6-way intersections.

A. Metrics

For the evaluation of our model, we have chosen two metrics: cumulative waiting time and average queue length of the vehicles in each episode.

Cumulative Waiting Time: The cumulative waiting time metric represents the total waiting periods experienced by vehicles at an intersection during a simulation. It is calculated by summing the waiting times of all vehicles from when they enter the intersection's influence zone until they clear it, reflecting the efficiency of the traffic signal control system.

Average Queue Length: The average queue length metric indicates the mean number of vehicles queued at an intersection per simulation step. It is calculated by dividing the total number of queued vehicles by the number of simulation steps, offering insights into congestion levels and the effectiveness of the traffic signal system in managing vehicle accumulation. A lower value suggests better congestion management and smoother traffic flow.

B. Comparison with traditional traffic signal control systems

We will now compare our model with existing traffic signal control systems. In traditional systems, the traffic light cycles through different phases with fixed durations. Based on real-world statistics, the green signal phases for N-S and E-W traffic last for 30 seconds each, while the left turn phases for N-S and E-W traffic last for 15 seconds each. The yellow light phase for each transition lasts 5 seconds. This data is based on U.S. traffic conditions, as we were unable to obtain official data for Indian roads.

TABLE I: Comparison Between Proposed Model And Existing Traffic Control Systems

Metrics	***Proposed DQN Model***	***Traditional Fixed Timings Systems***
Cumulative Delay	24651	33781
Average Queue Length	4.88	6.25

As shown in the Table I, our agent performs significantly better compared to the traditional system in both metrics: average queue length and cumulative waiting time. Specifically, it performs 27% better in cumulative waiting time and 22% better in average queue length. The simulation settings are the same for both cases: 1500 cars are generated randomly with a Weibull distribution.

To illustrate the performance of our model under varying traffic loads, we conducted tests comparing our model with traditional systems by increasing the traffic load from 110% to 150%. As demonstrated in Fig. 3 and Fig. 4 , the proposed model consistently outperforms fixed-timing systems. The true advantage of our model becomes evident under significantly high traffic loads, where it maintains stable performance even in challenging conditions. Specifically, under 150% traffic load, our model achieves a 43% improvement in average queue length and a 46% reduction in cumulative waiting time, showcasing its effectiveness in adapting to dynamic traffic scenarios.

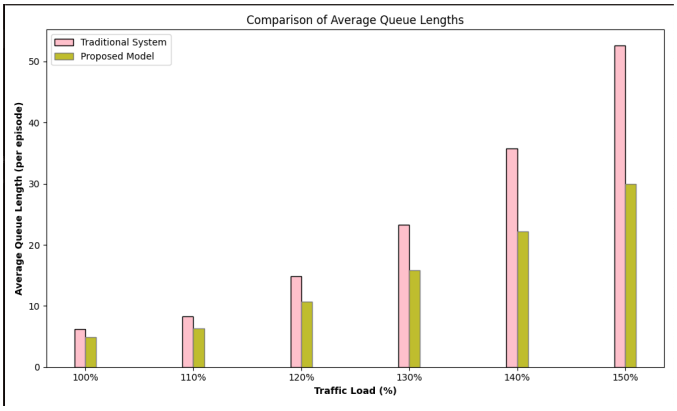
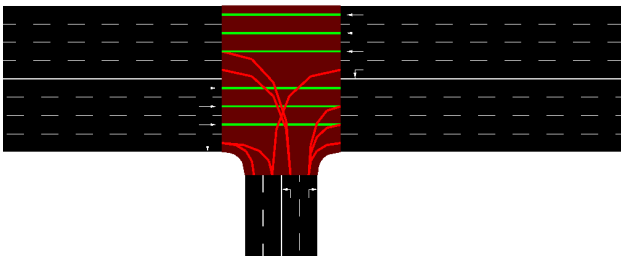


Fig. 3: Comparison of Average Queue Lengths under different traffic loads

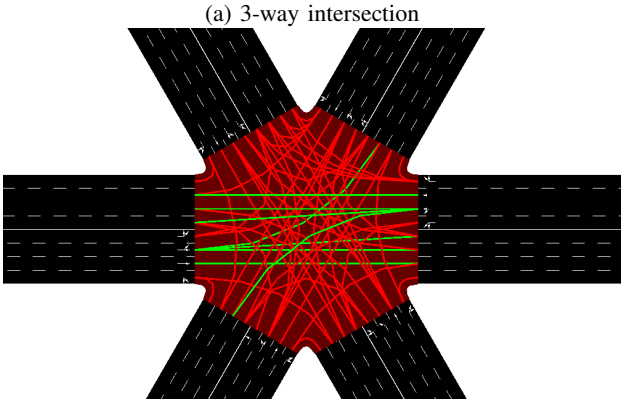
Fig. 4: Comparison of Cumulative Delay under different traffic loads

C. Performance on n-way intersections

The proposed model not only outperforms traditional systems in standard 4-way intersections but also demonstrates scalability to n-way intersections. To showcase its adaptability, the model was tested on 3-way 4-way, and 6-way (shown in Fig. 5) intersections. Adjustments were made to the number of states and actions to accommodate the different configurations: for 3-way intersections, the North direction was removed, reducing the number of states to 60 and actions to 3. In contrast, for 6-way intersections, the number of states increased to 180 and actions to 6. The networks, routes, and model architecture were updated accordingly.



(a) 3-way intersection



(b) 6-way intersection

Fig. 5: Networks for 3-way and 6-way intersections

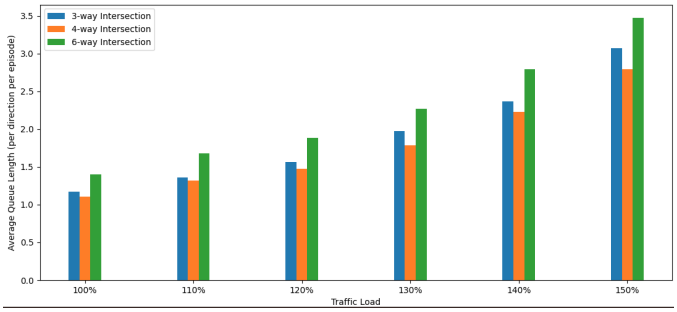


Fig. 6: Average Queue Lengths for different intersections under varying traffic loads

Fig. 6 illustrates the average queue lengths per direction for each type of intersection under varying traffic loads. The queue lengths are divided by the number of directions to enable a meaningful comparison. Although the model performs slightly less effectively in 3-way intersections compared to 4-way intersections due to a reduced action space, it still shows strong performance. The 6-way intersections, having the highest average queue lengths, reflect the challenge of managing more directions within a limited timeframe. Nonetheless, the performance differences remain manageable, demonstrating that the model can be effectively scaled with only minor reductions in performance.

V. Conclusion and Future Developments

This paper introduces an adaptive traffic signal control system using Deep Reinforcement Learning (DQN) integrated with SUMO. The system demonstrates significant improvements over traditional fixed-time controls by reducing cumulative waiting times and average queue lengths. This effectiveness highlights the potential of RL-based methods in optimizing traffic flow and managing congestion.

Future work can extend this approach by incorporating multi-agent systems for coordinated control across multiple intersections. Additionally, training with 3D images could enhance the model's performance by providing more detailed traffic data, leading to even better results. Overall, this method represents a promising advancement in adaptive traffic management, with significant potential for further development and application.

References

[1] L. Singh, S. Tripathi, and H. Arora, "Time optimization for traffic signal control using genetic algorithm," *Int. J. Recent Trends Eng.*, vol. 2, no. 2, pp. 4, 2009.

[2] F. V. Webster, Traffic Signal Settings, Report No. 39, 1958.

[3] P. Hunt, D. Robertson, R. Bretherton, R. Winton, Transport, and R. R. Laboratory, SCOOT: A Traffic Responsive Method of Coordinating Signals. *Transport and Road Research Laboratory*, 1981.

[4] J. Y. K. Luk, "Two traffic-responsive area traffic control methods: SCAT and SCOOT," *Traffic Eng. Control*, vol. 25, no. 1, 1984.

[5] J. Niittymäki and M. Pursula, "Signal control using fuzzy logic," *Fuzzy Sets and Systems*, vol. 116, no. 1, pp. 11-22, 2000.

[6] M. B. Trabia, M. S. Kaseko, and M. Ande, "A two-stage fuzzy logic controller for traffic signals," *Transportation Research Part C: Emerging Technologies*, vol. 7, no. 6, pp. 353-367, 1999.

[7] H. Ceylan and M. G. H. Bell, "Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing," *Transportation Research Part B: Methodological*, vol. 38, no. 4, pp. 329-342, 2004.

[8] M. Saito and J. Fan, "Artificial neural network–based heuristic optimal traffic signal timing," *Computer-Aided Civil and Infrastructure Engineering*, vol. 15, no. 4, pp. 293-307, 2000.

[9] D. Srinivasan, M. C. Choy, and R. L. Cheu, "Neural networks for real-time traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 261-272, 2006.

[10] T. Wang, Z. Zhu, J. Zhang, J. Tian, and W. Zhang, "A large-scale traffic signal control algorithm based on multi-layer graph deep reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 162, p. 104582, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X24001037

[11] D. Garg, M. Chli, and G. Vogiatzis, "A Deep Reinforcement Learning Agent for Traffic Intersection Control Optimization," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, 2019, pp. 4222-4229, doi: 10.1109/ITSC.2019.8917361.

[12] S. Alemzadeh, R. Moslemi, R. K. Sharma, and M. Mesbahi, "Adaptive Traffic Control with Deep Reinforcement Learning: Towards State-of-the-art and Beyond," arXiv:2007.10960, Jul. 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:220665679

[13] M. L. Puterman, "Markov decision processes," *Handbooks in Operations Research and Management Science*, vol. 2, pp. 331-434, 1990.

[14] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-Scale Application on Downtown Toronto," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, 2013.

[15] M. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[16] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic Traffic Simulation using SUMO," in *2018 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2018.