Traffic Light Signal Control using Deep Reinforcement Learning

Session Monsoon 2021

**Submitted by**

Sayantika Mandal

(1810110224)

Sanskar Tewatia

(1810110215)

**Under supervision**

**of**

Dr. Madan Gopal

(Professor, Electrical Department)

****

Department Of Electrical Engineering

School Of Engineering

Shiv Nadar University

(Spring 2021)

**Abstract**

An efficient transportation system is the need of the hour in developing countries, where incessant traffic congestions have become increasing resource-consuming, and pose an obstacle towards unceasing mobility. Traditional controllers such as Adaptive Traffic Signal Control (ATSC), based on sensors placed around traffic intersections and pre-defined fixed-time controllers optimized by traffic engineers, are unable to sufficiently resolve the traffic congestions since they do not account for the unexpected changes in traffic patterns. Though they are capable of regulating nominal traffic flow, they are vulnerable abnormal circumstances such as accidents, construction, and rush hours. Hence, an intelligent traffic light control is indispensable for alleviating the pressure caused due to traffic congestions. We propose an intelligent traffic light control agent, which will consider a four-way intersection as the environment, and use Deep Neural Networks to analyze the state representation of traffic such as queue length on the lanes, position of vehicles along with current and the next phase of traffic light via image representation from a simulator. The agent uses deep reinforcement learning techniques such as Deep Q-Networks (value-based method), REINFORCE (policy-based method) and Advantage Actor Critic method to suggest the optimal action of either changing or maintaining the current traffic light phase, so as to maximize the reward, which in our case is the average waiting time of vehicles and the queue length. In order to gather data to train and test the agent, SUMO (Simulation of Urban Mobility) simulator package is used, which helps emulate real-life traffic scenarios.

**Table of Contents**

List of figures …….………………………………………..……………………..  **(1)**

List of Tables ……………………………………………………………………..  **(2)**

1. Introduction ………………………………………………………………….. **(3)**
   1. Motivation ……………………………………………………………  **(3)**
   2. Reinforcement Learning ………………………………………………  **(3)**
      1. Types of Reinforcement Learning ……………….…………...  **(4)**
         1. Value-based methods ….…………………………........  **(4)**
         2. Policy-based methods ……………………………........  **(5)**
         3. Actor-Critic methods …………………………….........  **(5)**
      2. Q-Learning …………………………………………………...  **(5)**
      3. Deep Reinforcement Learning ……………………………....  **(5)**
   3. Problem Statement ……………………………………………………  **(6)**
2. Literature Review …………….………………………………………………  **(7)**
3. Work done ………………….…………………………………………………. **(9)**
   1. Experimental setting …….……………………………………………. **(9)**
   2. DRL Approach description ….….……………………………………. **(10)**
      1. State Representation ……………..……………….…………... **(10)**
      2. Action Set …………………………..…………….………….... **(11)**
      3. Reward Function ………………………………….…………... **(11)**
   3. Deep Q-Networks ….………………………………………………….. **(12)**
   4. Training Process ….……………………………………………….…... **(14)**
   5. Testing Results …....…………………………………………………... **(16)**
4. Plan for Next Half Semester ..….....…………………………………………... **(18)**
5. References …………………………………………………………………….  **(19)**

**List of Figures**

1.1 Reinforcement Learning cycle ..……………………………………………..  **(4)**

1.2 Q-Learning vs. Deep Q-Learning…..………………………………………..  **(6)**

3.1 SUMO Environment junction representing 2 types of traffic……………….. **(10)**

3.2 Image representation of a state of the environment………………………….. **(10)**

3.3 State representation of the east arm of the intersection..…………………….. **(11)**

3.4 Agent’s workflow………………………………...………………………….. **(12)**

3.5 DNN model structure ………………………………………………………..  **(13)**

3.6 Cumulative negative reward per episode during training…..………………..  **(15)**

3.7 Average queue length of vehicles per episode during training .……………..  **(16)**

3.8 Reward of agent per action step during testing……..………………………..  **(16)**

3.9 Queue length of vehicles per simulation step during testing…..…………….. **(17)**

**List of Tables**

3.1 DQN agent and DNN model hyper-parameter details ……………………..  **(14)**

**Chapter 1**

**Introduction**

**Motivation:**

With a population of over 7 billion and increasing, the vehicular traffic has always been on the rise. As more manufacturers are competing with each other to reduce the expenses of purchasing or owning vehicles, people today are in a better capacity of being able to own their own cars. Even in the presence of public transport such as underground metros, people prefer to travel in their own personal vehicles. This has led to a massive increase in the demand of efficient traffic management, be it traffic light control, or building efficient architectures that completely alleviate the need of traffic lights [1]. However, such construction activities require immense planning, budget allocation and might not be feasible in crowded areas where the space is also limited. Hence, there is a scope of optimizing traffic signal control, in order to better manage the traffic flow and reduce the waiting time as well as the queue length at various intersections throughout the world.

At present, traffic signals are controlled by either setting a fixed duration for each state of the traffic light [2, 3], or by using the vehicle-activated inductive sensors underneath the road, and using these inputs to estimate the traffic signal duration. However, both of these methods, though simple to implement, are unable to adapt to emergency, accidental or rush hour scenarios. In such tricky cases, usually traffic policemen have to step in, and manually control the flow of traffic, on the basis of the congestions they witness, based on their previous experiences of controlling such crowded junctions. Thus, an adaptive traffic light control technique is the need of the hour as such a technique would be able to accurately process the current state of traffic, and take optimal actions in regard to the reducing the congestions, and thereby decreasing the waiting time and queue length [4, 5, 6].

**Reinforcement Learning:**

Reinforcement Learning (RL), a feedback-based Machine Learning technique, is the most-used methodology for traffic signal control and has been shown to have the potential to dynamically adjust traffic lights according to real-time traffic in recent studies. In a Reinforcement Learning problem, an agent observes a stochastic environment, perceives its state and learns automatically by exploring the environment and performing actions on a timely basis which results in a change in the environment to another state [7]. A policy function governs the transitions between states in a given time step as defined by the Markov Decision Process (MDP) which describes an environment in RL [8]. The agent obtains feedbacks on the basis of the actions taken, known as the reward which acts as performance measure. A positive reward is the result of a good action, whereas a bad action leads to a negative reward or penalty.

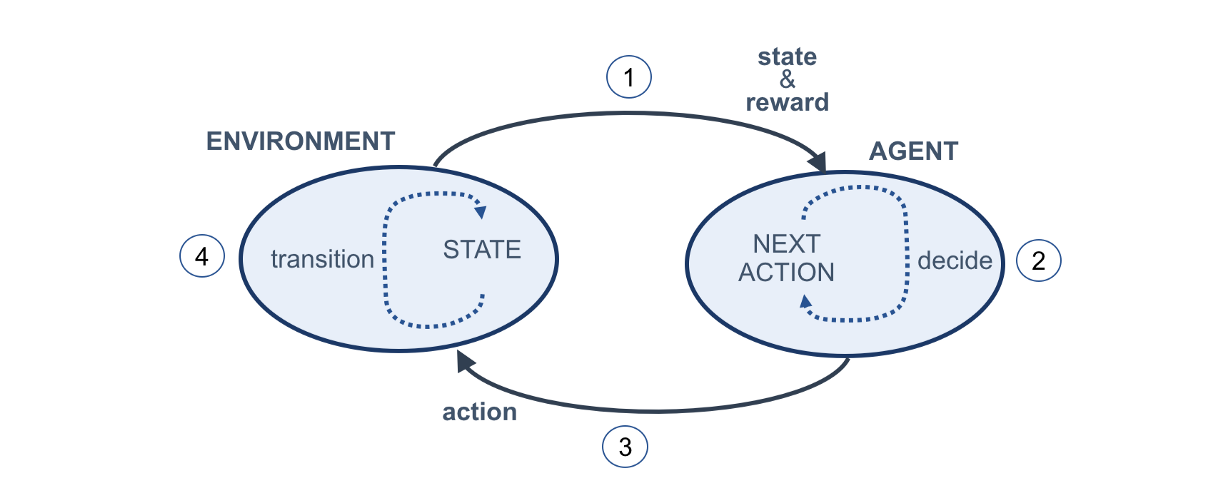


Fig. 1.1: Reinforcement learning cycle

The agent’s main goal is to learn the optimal policy defining the behavior of the agent in order to maximize the cumulative reward. At the start of the learning process, the agent performs random actions, indication that the control policy is not known. However, as the agent explores the state-action space extensively and learns through time, it starts exploiting the learned policy and the tendency of choosing random actions declines. The Exploration vs. Exploitation trade-off is the term coined for the dilemma faced by the agent in the exploration of the new states while maximizing its overall reward simultaneously. The Epsilon-Greedy algorithm is the most used method to balance exploration and exploitation and performs actions by choosing between exploration and exploitation randomly [9]. Epsilon refers to the probability of choosing to explore and the algorithm chooses to exploit most of the time with a small chance of exploring.

1. **Types of Reinforcement Learning:** There are three types of RL methods: value-based, policy-based and actor-critic methods [10].
   1. ***Value-based methods:*** Value-based approach has the goal of finding an optimal value function, which is the maximum value at a state under any policy and maximizing the expected future rewards. Q-learning is a value-based RL method [11].
   2. ***Policy-based methods:*** Policy-based approaches such as REINFORCE, directly approximate the optimal policy for the maximum future rewards without using the value function, through policy iteration. [12].
   3. ***Actor-Critic methods:*** Actor-critic integrates both value and policy-based approaches in a single RL framework, where the critic estimates the value function and the actor estimates the best policy [13] This approach uses two separate models for each of its parts.
2. **Q-Learning:**  Q-learning[14] is an off-policy method in which the value learnt by the agent is based on an action derived from another policy. It is the most used model-free RL algorithm for traffic control [15]. The method updates the optimal Quality (Q)-value of each state-action pair iteratively according to the Bellman equation and stores in a Q-table in agreement with the respective state-action pairs, as shown in Figure 1.2. These Q-values are representative of the expected future rewards for each state-action pair.
3. **Deep Reinforcement Learning:** Deep Reinforcement Learning (DRL) is the combination of RL and Deep Learning (DL). In the case of traffic control systems, the state-action space is extremely huge as the environment is very complex and hence the Q-table may suffer from high-dimensionality. Thus, as the number of states increases, the amount of memory required to save and update the Q-table would increase and the amount of time required to explore each state to create the required Q-table would be unrealistic. As a solution, DL models such as Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN) are used to approximate the Q-value efficiently, which eradicates the need for maintaining a Q-table in the RL framework. This concept is also known as Deep Q-Learning (DQL). Deep Q-Network (DQN) algorithm as shown in Figure 1.2, modifies Q-learning and uses a technique called Experience Replay (ER), in which the agent’s experiences are stored at each time step in a data set combined over many episodes into a replay buffer memory [16].

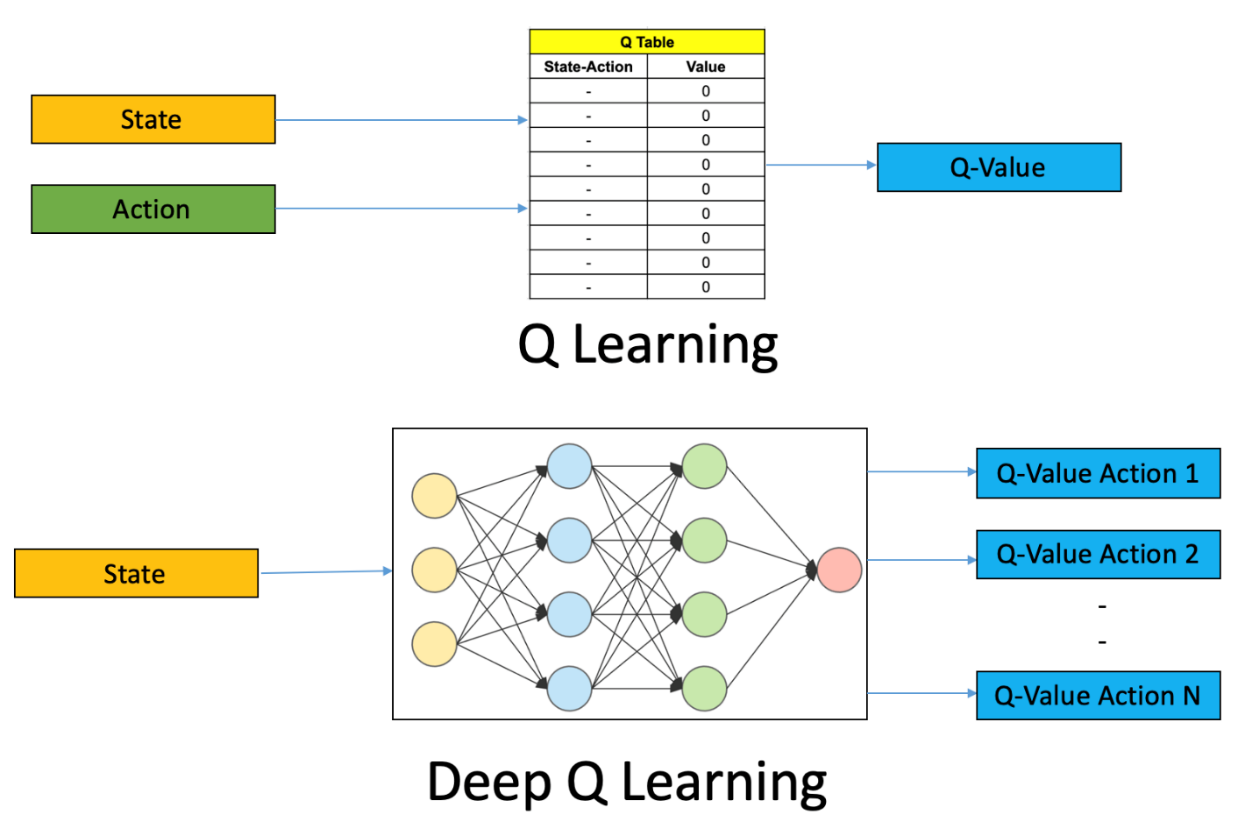


Fig. 1.2: Q-Learning vs. Deep Q-Learning

**Problem Statement:**

The objective of this project is to build a Deep Reinforcement Learning traffic light control agent to manage a traffic-lights regulated single four-way intersection as the environment provided by the Simulation of Urban Mobility (SUMO) simulator. The agent learns DRL algorithms to estimate the reward using Deep Neural Networks to perceive the current state of the traffic via images from the simulator. It then chooses the optimal action regarding the current traffic light phase from a fixed set of predetermined actions, based on the exploitation of its learning experience in order to maximize the reward and as a result, optimize the intersection's traffic efficiency.

**Chapter 2**

**Literature Survey**

**Reinforcement Learning related works:**

Reinforcement learning has been used to successfully fulfil several challenging tasks in a variety of disciplines, including games [3], robotics [4], and traffic signal management. Various possible approaches using different algorithms and neural network structures have been proposed in the recent works on Reinforcement Learning to solve the traffic signal control problem.

IntelliLight, an intelligent traffic control system implemented by H. Wei et al [1], used a Deep Q-network framework to train the DRL agent. To resolve the difficulties in distinguishing the decision process for different traffic light phases, the study proposed a special sub-structure known as Phase Gate and also utilized the Memory Palace theory to improve the fitting capability of the network to predict the reward accurately. On real-world data, their approach IntelliLight is able to obtain the best reward, queue length, delay, and duration among baseline methods. K. Tan et al [17] also utilized the Deep Q-network algorithm to test their DRL-based adaptive traffic signal control framework that explicitly considers realistic traffic scenarios, sensors, and physical constraints. In this framework, they also proposed a novel reward function that shows significantly improved traffic performance compared to the typical baseline pretimed and fully-actuated traffic signals controllers. Their DRL agent’s performance was tested on real traffic data during high traffic demand periods which outperformed both the baseline controllers. D. Li et al [16] built a truly adaptive traffic signal control model in a traffic simulator using the technology of modern deep reinforcement learning. The model proposed was based on Deep Q-network algorithm and both single-agent (for single 4-way intersection) and multiagent (for multiple intersections) cases were demonstrated. They tested with data sets pertaining to three different traffic conditions, and proved that their proposed model is better than other methods (e.g., Q-learning method, longest queue first method, and Webster fixed timing control method) for all cases. The proposed model reduces both the average waiting time and travel time, and it becomes more advantageous as the traffic environment becomes more complex.

A. Vidali et al [18] in their research work, adopted the Deep Q-learning mechanism to train two DRL agents using two different reward functions and discussed the performance of both agents with a static traffic light benchmark. The cumulative wait time and the average wait time per vehicle were the metrics used to assess the difference in the performance of the two agents. Sajad et al [7] proposed two kinds of reinforcement learning algorithms: deep policy-gradient and value-function based agents to predict the best possible traffic signal. In their policy-gradient agent, there is mapping of observations to the control signal whereas in the value-functioned agent, first the values are estimated for all legal signals, and the optimal one is selected. They performed gradient descent on the policy parameters. In the value-functioned based approach, they used deep neural networks to estimate the action-value function, which maps the input state to the action values (i.e., the future reward that can be achieved for that given state and action). Liang et al [9] used data collected from different sensors and vehicular networks and proposed a model with prioritized experience replay. The general idea is that they extend the duration for the phase that has more vehicles in that direction, and hence this adaptive phase duration optimization is their key problem. They use state of the art techniques along with CNN and propose a network termed Double Dueling Deep Q Network.

Though all the above-mentioned studies learn a Q-function to map state and action to reward, they however differ in the state representation including features such as queue length [1, 19, 20], number of vehicles [1], average delay [1, 5, 21], average travel time [17] and features based on image representation of vehicles’ position [1, 5, 9, 18, 22, 23]. The reward functions chosen also vary, containing average delay [1, 4, 7, 17, 22, 24], average travel time [5, 22], queue length [1, 17, 19], total waiting time [18].

**Chapter 3**

**Work Done**

**Experimental Setting:**

SUMO was used as the traffic micro-simulator for this project [25]. Simulation of Urban Mobility (SUMO) is a free and open-source traffic simulation software which includes an infrastructure editor and an application programming interface (API) and offers advanced features such as multimodal traffic, route generation, traffic light control, and a multitude of customizable road network maps. This software has been widely used to simulate all sorts of traffic conditions, and is used to optimize road networks in order to improve the flow of traffic, and reduce the waiting times. Results gathered from the simulations conducted in this software could be used to construct efficient traffic networks.

In particular this software was chosen because it can be interfaced directly from the command line and through python by using the TraCI library. Using this library, one can change all the parameters of the network, control traffic lights, and retrieve various observations in regard to the flow of traffic, and using these values to tune and manipulate the parameters instantly. In our project, first we created an empty road network by specifying all the nodes, junctions, traffic lanes, and the traffic light. Then we created a separate python file to generate traffic on this road, specifying the total number of cars and then bonded them together to create what is called a SUMO config file, which has moving traffic, and can be observed in real time.

Up till now, the easiest junction possible was chosen in order to create a baseline model. This junction consists of only two types of traffic – either East-West traffic or North-South traffic as shown in Figure 3.1. Each road has 2 lanes, and these roads have lengths equal to 750 metres, with the junction in the middle. The duration of green signal is chosen to be 8 seconds, and the duration of yellow signal is 3 seconds. These values were chosen after extensive testing and tuning, and the TraCI library can control whether to change the traffic signals or not.

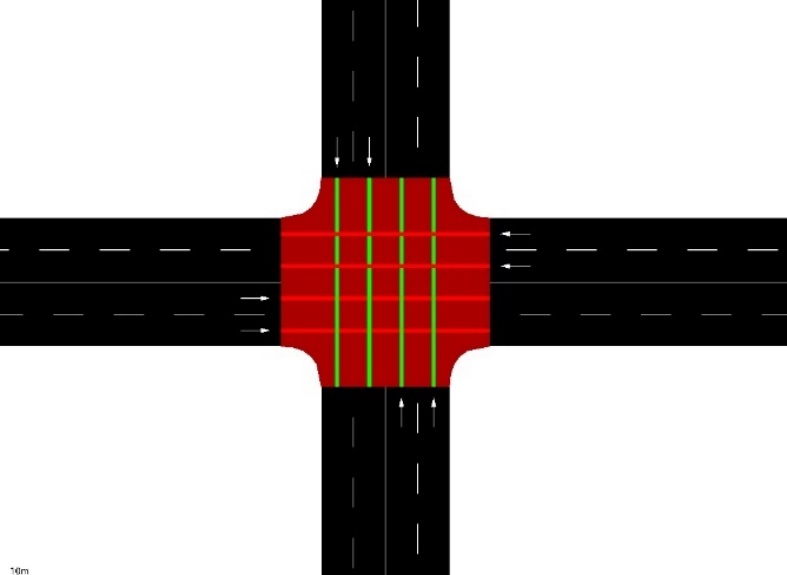


Fig. 3.1: The SUMO environment junction representing 2 types of traffic

**DRL approach description:**

The reinforcement learning framework we used is defined as follows:

1. **State representation:** The representation of the traffic situation in the environment in a given action step t is denoted by st. The spatial information of the vehicles inside the environment was used as the state representation as seen in Figure 3.2.

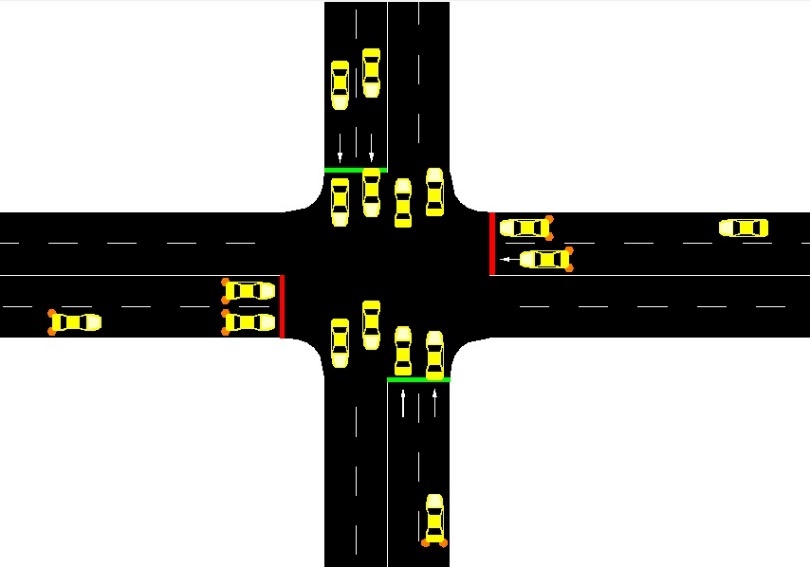


Fig. 3.2: Image representation of a state of the environment

The continuous environment was made discrete by using cells to represent the presence and position of the vehicles on the lanes. As shown in Figure 3.3, each arm of the intersection was discretized into ten cells and the state is represented as a binary matrix wherein the presence of a vehicle in a cell is denoted by a ‘1’ and the absence denoted by a ‘0’. Since movement direction of vehicles in both the lanes is straight, the ten cells cover both the lanes. Therefore, there are 40 cells in the entire intersection and hence the number of states is 40. Also, the cells aren’t the same size: the cell length increases as its distance from the intersection increases because the main motive is to detect standing vehicles near to the intersection. This also decreases the computational complexity which would have arisen in the case of equal and short sized cells. The length of the shortest cells is 7 meters which is 2 meters extra than the length of the cars.

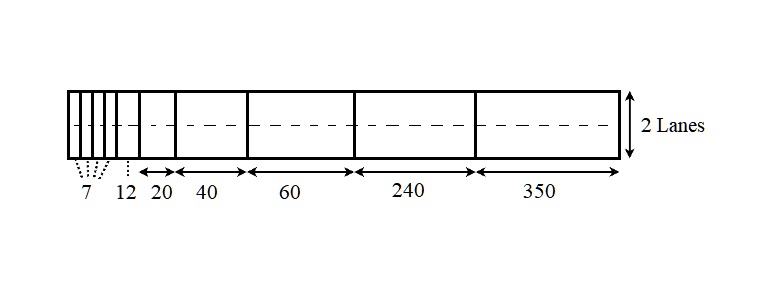


Fig. 3.3: State representation design of the east arm of the intersection

1. **Action Set:** The set of 2 possible actions that the agent can choose from is defined in (1).

North-South Green (NSG) means that the green traffic phase is active for vehicles in the north-south intersection arm which want to go straight whereas in East-West Green (EWG) the green traffic phase is active for vehicles in the east-west intersection arm. However, if the two actions chosen at consecutive action steps are the same, then a yellow phase is introduced in between.

1. **Reward Function:** The reward function uses the accumulated total waiting time as the metric as defined in equations (2) and (3).

Where v represents a vehicle, n is the total number of vehicles in the environment at action step t, and awt(v, t) is the amount of time (in seconds) a vehicle v has a speed of less then 0.1m/s at action step t, since its arrival in the environment. In equation (3), r(t) is the reward at action step t, and atwt(t) represents the accumulated total waiting time of all the vehicles in the intersection at action step t.

Figure 3.4 shows the workflow of the DRL agent in the form of a flow chart.

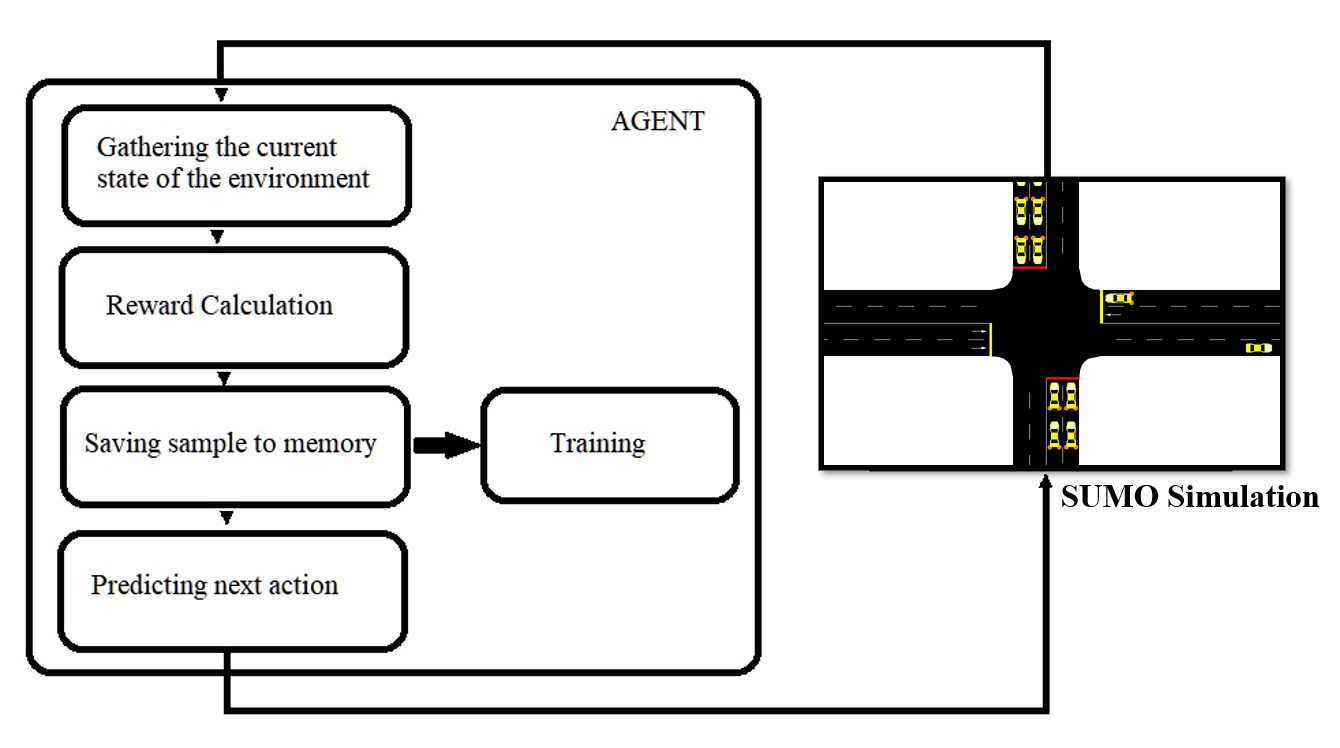


Fig. 3.4: Agent’s workflow

**Deep Q-Networks:**

The agent learned a Deep Q-Network (DQN) to predict the reward. In other words, the learning mechanism used here was Deep Q-Learning (DQL), which is an augmentation of the value-based Q-learning algorithm. In Q-learning, the Q-value is updated by using the newly received sample (st, at, st+1, rt) according to the Bellman equation as described in equation (4).

where is the action at taken from state st at action step t or the current Q-value of state–action pair, is a new Q-value, α is learning rate, rt is the reward associated to taking action at from state st, received from the environment, γ is discount rate (0 < γ < 1) which can be static or change over time in order to model a process in which the earlier rewards are worthier than the future rewards, is the estimate of optimal future value and a is any action from the possible action space. The term () is known as the learned value.

Since we used DQL, the new Q value was approximated by the DNN model implemented with weights θ and is represented as . The DNN model is learned to minimize the expected squared error (loss) between the predicted Q-value and the target Q-value, as described by the following equation (5):

A fully-connected DNN was built with an input layer of 40 neurons (as there are 40 state representations), 5 hidden layers with varying number of neurons each using Rectified Linear Unit (ReLU) as the activation function and the output layer with 2 neurons with linear activation function as there are 2 possible actions given a state. The DNN model weight-updating was done based on the Adam optimization method in every learning iteration to induce the minimization of equation (5). The DNN model structure is shown in Figure 3.5.

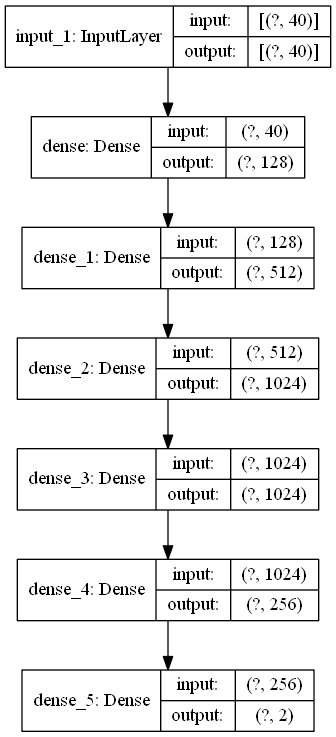


Fig. 3.5: DNN model structure

Table 3.1 summarizes the hyperparameter values of the DQN agent and the DNN model as used in this work.

Table 3.1: DQN Agent and DNN model hyper-parameter details

|  |  |  |
| --- | --- | --- |
|  | **Parameter** | **Value** |
| DQN Agent | Number of states | 40 |
| Number of actions | 2 |
| Discount rate (γ) | 0.95 |
| Learning rate (α) | 0.0001 |
| DNN Model | Depth | 5 |
| Batch size | 100 |
| Number of epochs | 800 |
| Learning rate | 0.0001 |

**Training Process:**

The entire training was done across 150 episodes and the period of each episode was set at 2 hours. Since the time frequency in SUMO is 1 second per step, 1 simulation step corresponds to 1 second and hence the maximum number of steps per episode is 7200. We arrived at these values by trial-error method by starting the training from 100 episodes of half hour each and then 125 episodes of 1 hour each and comparing the results at the end. In each episode the traffic was generated according to a Weibull distribution with a shape of 2 in order to maintain a high degree of reality. Every generated vehicle had similar physical dimensions and performance in terms of acceleration and speed, and a total of 1250 vehicles were generated for the training purpose.

The technique of Experience Replay was adopted during the training phase where the agent gets randomized samples in the form of batches from a buffer memory for the purpose of learning as demonstrated in Figure 3.4 above. This improves the learning efficiency and performance of the agent. The memory size which represents the maximum number of samples that can be stored, was set at 50000 samples. Further, the most used ɛ-greedy exploration policy was employed for action-selection where the value of ɛ was set according to equation (6).

where n is the current episode of training and N in the total number of episodes. The algorithm explores with a probability ɛ and initially ɛ = 1, indicative of the agent exploring. However, with the progress in training, ɛ keeps decreasing and the agent exploits its learning experience with a probability 1- ɛ.

The performance of the agent was analyzed on the basis of the reward curves during the training and the average queue length (sum of the number of waiting cars) for every episode. Figure 3.6 shows the learning improvement during the training in terms of cumulative negative reward which is the magnitude of the negative outcomes of the actions during each episode. The reward curve is not significantly unstable, indicating that the agent was able to learn a sufficiently correct policy.

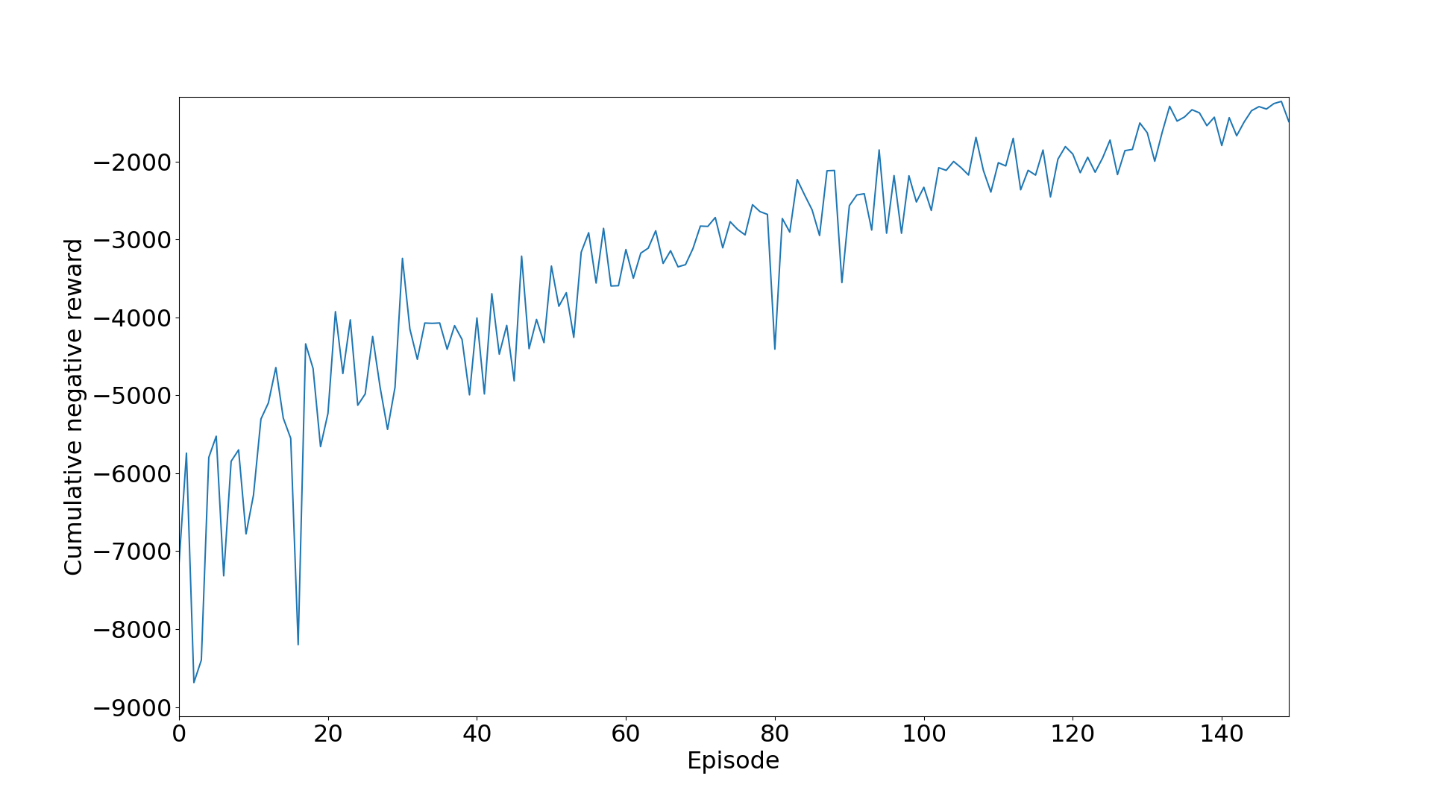


Fig. 3.6: Cumulative negative reward per episode during training

Figure 3.7 displays the curve of average queue length of vehicles as the training proceeds and the decreasing nature of the curve indicates that the agent was able to minimize the number of waiting cars at the junction and clear traffic congestion.

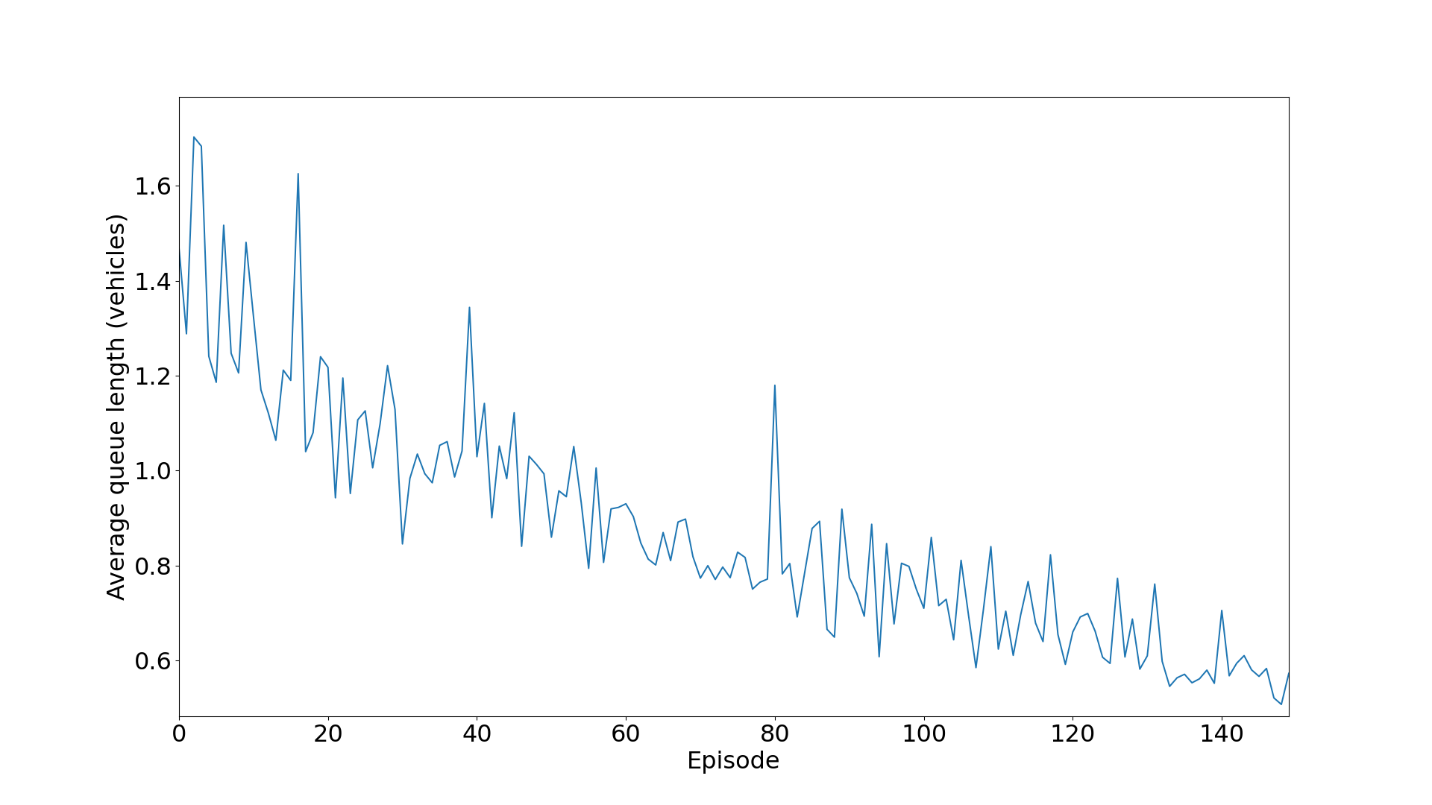


Fig. 3.7: Average queue length of vehicles per episode during training

**Testing Results:**

The results obtained after testing the agent are shown in Figures 3.8 and 3.9 representing the reward and queue lengths at each step.

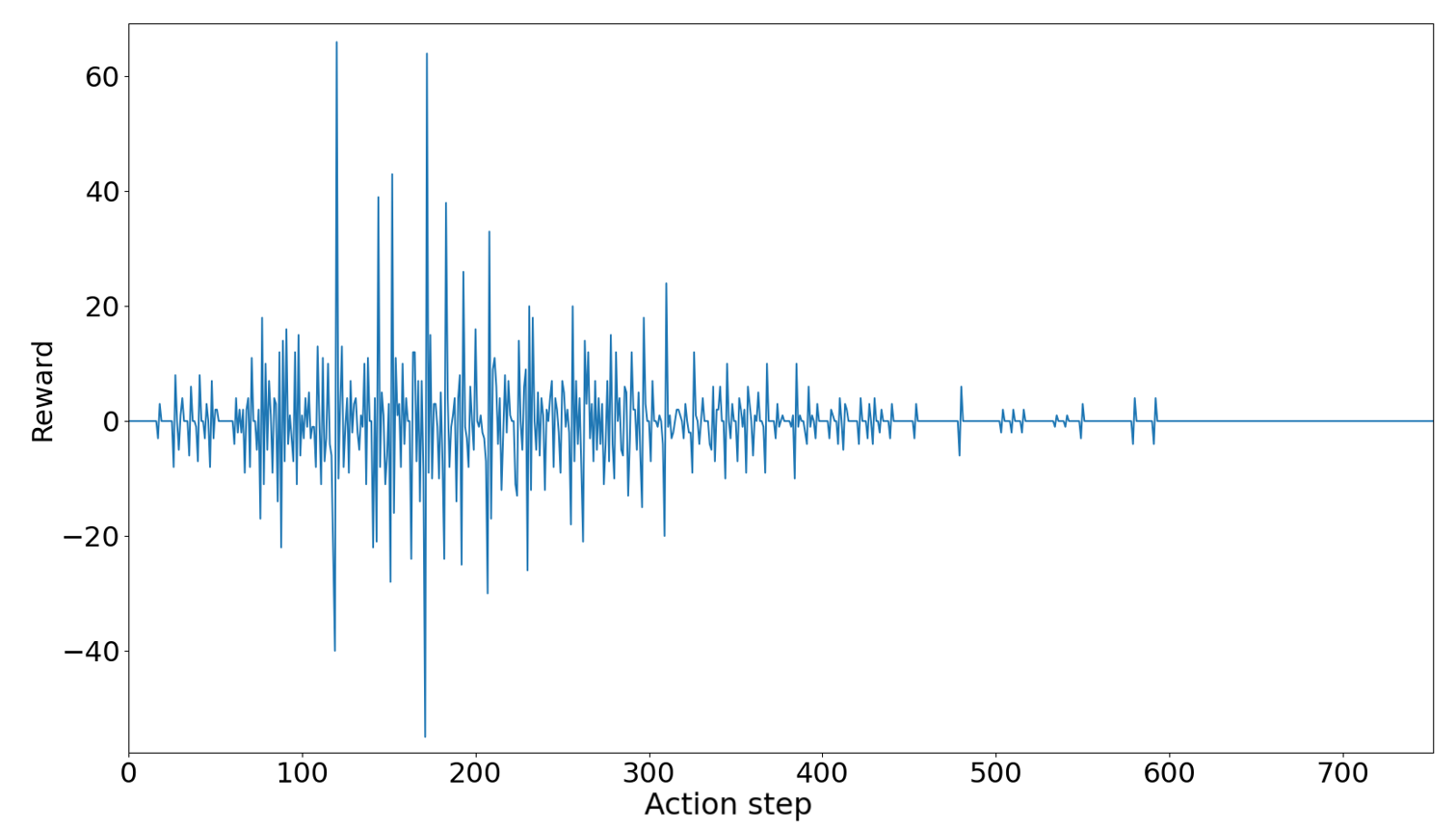
****

Fig. 3.8: Reward of agent per action step during testing

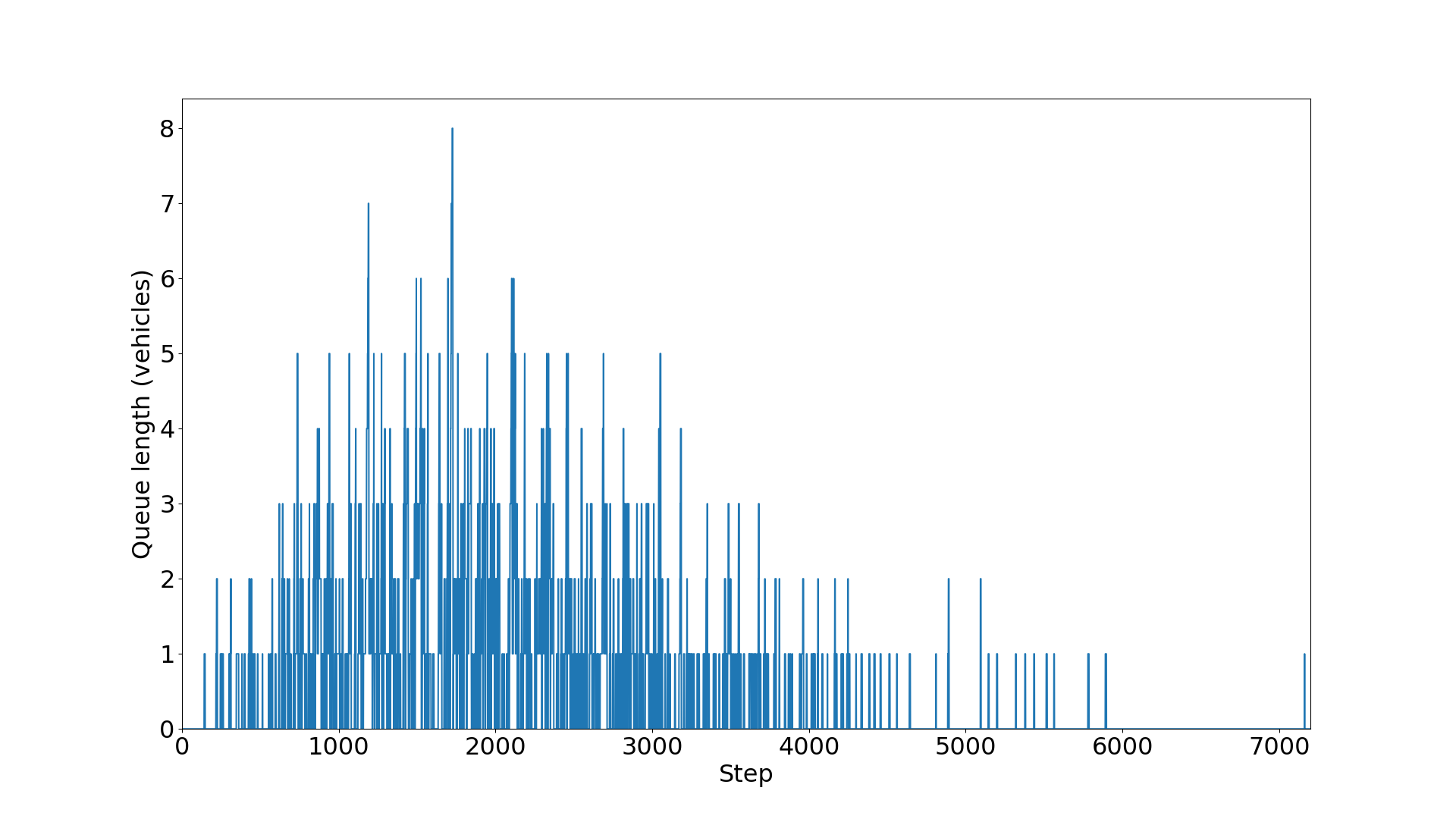
****

Fig. 3.9: Queue length of vehicles per simulation step during testing

**Chapter 4**

**Plan for Next Half Semester**

**Future Work:**

Till now the experimentation was done on the simplest kind of intersection, with traffic going only in one direction, and hence only 2 lanes were required for this. In the remaining half of the semester, there is a possibility for us to try and experiment on more complex architectures of intersections, where the vehicular traffic is free to switch direction. For example, a car coming to the intersection from the southern direction might want to change its direction and go to the eastern or western direction. Such cases will require careful planning of the number of lanes, setting appropriate amount of turning lanes, deciding the chronological order in which the traffic signals will be actuated.

The duration of the signals will be modified as well as fixing the right-hand and left-hand turn lanes for each direction. The kind of traffic that we generated right now is only North-South or East-West. In the above scenario, we will have to appropriately divide the number of cars which might want to change directions, or the ones which will continue to go straight after passing through the intersection.

Furthermore, we will try to implement other DRL methods such as REINFORCE and Advantage Actor-Critic (A2C) method as the learning mechanism for the agent and compare the obtained results.

**References**

1. H. Wei et al, “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, July 2018.
2. Alan J Miller. 1963. “Settings for fixed-cycle traffic signals.”, Journal of the Operational Research Society 14, 4 (1963), 373–386.
3. F. V Webster. 1958. Traffic signal settings. Road Research Technical Paper 39 (1958).
4. Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. “Multiagent reinforcement learning for urban traffic control using coordination graphs.”, Machine learning and knowledge discovery in databases (2008), 656–671.
5. Elise van der Pol and Frans A Oliehoek. 2016. Coordinated Deep Reinforcement Learners for Traffic Light Control. NIPS.
6. Wiering, M.,“Multi-agent reinforcement learning for traffic light control”. In Machine Learning: Proceedings of the Seventeenth International Conference (ICML’2000), pp. 1151–1158, 2000.
7. Sajad Mousavi, S., Schukat, M. and Howley, E., “Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning,” arXiv e-prints, pp. arXiv-1704, 2017.
8. Martin et al., “Application of Deep Reinforcement Learning in Traffic Signal Control: An Overview and Impact of Open Traffic Data”, Journal of Applied Sciences, vol. 10(4011), 2020.
9. Liang, X., Du, X., Wang, G. and Han, Z., “Deep reinforcement learning for traffic light control in vehicular networks,” arXiv preprint, arXiv:1803.11115, 2018.
10. Sutton, R. S., and Barto, A. G., 2018. “Reinforcement learning: An introduction”.
11. Watkins, C. J., and Dayan, P., 1992. “Q-learning”. Machine learning, 8(3-4), pp. 279–292.
12. Williams, R. J., 1992. “Simple statistical gradient following algorithms for connectionist reinforcement learning”. Machine learning, 8(3-4), pp. 229–256.
13. Konda, V. R., and Tsitsiklis, J. N., 2000. “Actor-critic algorithms”. In Advances in neural information processing systems, pp. 1008–1014.
14. C. J. Watkins and P. Dayan, “Q-learning,” Machine learning, vol. 8, no. 3-4, pp. 279–292, 1992.
15. Chin, Y.K.; Kow, W.Y.; Khong, W.L.; Tan, M.K.; Teo, K.T.K. Q-Learning Traffic Signal Optimization within Multiple Intersections Traffic Network. In Proceedings of the 2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation, Valetta, Malta, 14–16 November 2012; pp. 343–348.
16. Li, D.; Wu, J.; Xu, M.; Wang, Z.; Hu, K. “Adaptive Traffic Signal Control Model on Intersections Based on Deep Reinforcement Learning,” J. Adv. Transp. 2020, 2020.
17. K. Tan et al, “Deep Reinforcement Learning for Adaptive Traffic Signal Control”, Proceedings of the ASME 2019, Dynamic Systems and Control Conference DSCC2019, November 2019.
18. A. Vidali, L. Crociani, G. Vizzari, and S. Bandini, ‘‘A deep reinforcement learning approach to adaptive traffic lights management,’’ in Proc. Workshop ‘From Objects Agents’, pp. 42–50, 2019.
19. L. Li, Y. Lv, and F.-Y. Wang, “Traffic signal timing via deep reinforcement learning,” IEEE/CAA Journal of Automatica Sinica, vol. 3, no. 3, pp. 247–254, 2016.
20. Patrick Mannion, Jim Duggan, and Enda Howley. 2016. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In Autonomic Road Transport Support Systems. Springer, 47–66.
21. W. Genders and S. Razavi, “Using a deep reinforcement learning agent for traffic signal control,” arXiv preprint arXiv:1611.01142, 2016.
22. Mengqi Liu, Jiachuan Deng, Ming Xu, Xianbo Zhang, and Wei Wang. 2017. “Cooperative Deep Reinforcement Learning for Tra ic Signal Control.” (2017).
23. Gao, J., Shen, Y., Liu, J., Ito, M., and Shiratori, N., “Adaptive traffic signal control: Deep reinforcement learn learning algorithm with experience replay and target network”. arXiv preprint arXiv:1705.02755, 2017.
24. Itamar Arel, Cong Liu, T Urbanik, and AG Kohls. 2010. Reinforcement learning-based multi-agent system for network traffic signal control. IET Intelligent Transport Systems 4, 2 (2010), 128–135.
25. D. Krajzewicz, G. Hertkorn, C. R¨ossel, and P. Wagner, “Sumo (simulation of urban mobility)-an open-source traffic simulation,” in Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002), 2002, pp. 183–187.