

# Traffic Light Control Using RFID and Deep Reinforcement Learning



Shivnath Yadav, Sunakshi Singh, and Vijay Kumar Chaurasiya

**Abstract** Roads are the indispensable need of our modern society. From connecting places to exporting goods, it registers as one of the important modes of transportation. Nevertheless, existing infrastructure has some issues such as congestion, delay, and energy wastage. Therefore, to address these aforementioned concerns, there are two most probable solutions. One way is to simply expand the transportation capacity by building road systems that can handle extensive traffic flow. Another way is to operate on the existing infrastructure by improving the systems that have a significant impact on the traffic flow, such as traffic signal controllers. This approach is expedient considering low cost in implementation and reuses the current equipment. Therefore, in this chapter, we have proposed a four way-intersection which uses a feedforward neural network and the Q-learning algorithm to get value function approximation. We have tried to improve the traffic flow at the intersection point which is monitored, managed, and controlled by traffic lights. For achieving such a goal we will employ deep reinforcement learning commonly called deep Learning. To get real-time data, RFID readers will be used. The agent will be given the responsibility to manage the traffic light's phase activation so as to optimize the traffic efficiency. The agent decision is based on the analysis of the deep Q-learning algorithm. In order to choose the best light phase, deep Q-learning algorithm divides each road of intersection in the form of a one-dimensional array and calls each section as a state. Reward is awarded on the basis of the duration of the waiting time of a vehicle. Simulation is done on the SUMO tool (Simulation of Urban Mobility) to evaluate the model. The result is added at the end which clearly shows the efficiency of the model compared to others.

**Keywords** Traffic light control · Deep Q-learning · Artificial neural networks · RFID

---

S. Yadav · S. Singh (✉) · V. K. Chaurasiya  
Indian Institute of Information Technology, Devghat, Jhalwa, Prayagraj 211015, Uttar Pradesh, India

# 1 Introduction

Population growth and urbanization had sped up the demand for a good transportation system. On the other hand, the existing infrastructure roads and its corresponding management are not satisfactory to bear the increasing traffic volume [1]. This had led to several problems like congestion which causes delays and subsequently had an impact on the environment through noise and air pollution [2]. So proper traffic management is a must. There were numerous research ideas that were put forward however, most of them consist of a set of codes (programs). These programs were not only computationally complex but were based on certain assumptions. As a consequence, they were not effective in the real-time scenario. Some researchers tried to employ machine learning algorithms like fuzzy logic, genetic logic algorithm, etc. Still, these methods failed to achieve the desired output because they were not considering the real-time data inputs. Therefore, a good traffic system is required which should do the continuous monitoring of the traffic in real-time and take the decision after analysing all the roads at the intersection.

In order to achieve the above goal, we are going to employ an RFID reader for continuous monitoring (assuming that every vehicle is having an RFID tag). For analysing and decision-making we will take the help of reinforcement learning [3, 4]. The reason behind using a reinforcement learning technique is that it does not depend upon the heuristic assumptions and equations and fits perfectly in the model to master the optimal control through the previous experiences of handling the traffic. The concept of reinforcement learning is based on the Markov decision process [5]. It has been extensively used as a computational tool. The traditional reinforcement learning algorithm was less optimal with limited scalability. So, Q-learning, a popular RL algorithm, and a deep neural network was chosen because of its better learning capacity [6, 7].

The use of information and communication technology to improve the quality of living is referred to as smart cities. As a result of continual development and population growth, smart cities have evolved. Smart cities seek to address these issues in order to improve inhabitants' quality of life. When a city can develop and implement creative solutions that are based on cutting-edge technologies and cutting-edge scientific knowledge, it is considered smart. To put it another way, a city gets "smarter" by developing and implementing data-driven solutions to make it easier to monitor, understand, analyse, plan, and optimize its operations, activities, services, and policies.

## A. Outline of Chapter

The remaining portion of the chapter is tabulated in the following order: Sect. 2 gives the synopsis of the existing research work of traffic light control. While Sect. 3 outlines the preliminaries. Section 4 describes the learning mechanism of the model and describes its corresponding state, action, and reward in detail. Section 5 discusses the experimental setup and training of the whole model additionally an analysis and the performance of the results are also mentioned. Finally, the conclusion and future work are described in Sect. 6.

## 2 Related Work

In this paper [8] the author put forward an adaptive traffic signal control system that was trained by deep Q-learning and reduces the travel time by 20% and waiting time by 82%. In another attempt, the author tried to reduce the waiting time of the vehicle by using a multi-agent reinforcement learning algorithm. Furthermore, they showed their model outperformed even after an increase in traffic load.

In the paper [9], the author has constructed a two-way street model with a controllable traffic light which is dependent on the time function each cycle time is different from the previous one. While in [8] the author applied a deep reinforcement learning method to propose an adaptive traffic signal control system (simulator) and trained the model using Q-learning. In [10] paper Juntao Gao et al. modelled a deep reinforcement learning algorithm that takes the attribute from the real-time data and masters the excellent policy for adaptive signal control. In this research paper, [11] author attempted to adjust the duration of traffic lights dynamically. For that, they divided the whole scenario into smaller sections and employed conventional neural network to map the reward with the state. The state was defined by vehicle position and speed information.

Centralized reinforcement learning is not feasible for adaptive traffic signal control. This paper [12] presents a fully scalable and decentralized MARL algorithm. Proposed novel A2C-based MARL. The author put forward an intelligent driving model in [13] that functions on the input of this adaptive traffic signal control which employs a multi-agent framework that dynamically learns the behaviour of the driver. They have used the Bayesian interpretation of probability for decision-making. This paper [14] put forward a load balancing approach that relies on the prediction of the traffic situation. This is achieved by the efficient cooperation of the micro base stations. Spatiotemporal correlation with CNN is used to predict the traffic situation. In this paper [15] author integrated deep learning with the Bayesian model (proposed by Wang) to predict the traffic flow.

One advantage is that it overcomes the error magnification phenomenon. In another paper [16] employs deep learning. The model was specifically designed to learn traffic speed. In order to predict the real-time traffic speed, they employed LTE (Long Term Evolution) data.

This paper [17] presents an efficient scheduling method. The proposed method is able to control a dynamic and complex traffic environment with the help of reinforcement learning techniques. In another paper [18] author designed an adaptive signal control system and called it RHODES. For the smoother working of the adaptive signal control, real-time information is taken from the detector. In order to predict the arrival of vehicles at the intersection point, the author had used the PREDICT algorithm which was proposed by Head. For that, this algorithm uses detectors output, traffic state, and planned phase timing on reaching each upstream intersection. This adaptive model could predict both the short-term and medium-term fluctuations of the traffic. This approach could even set phases so as to maximize performance.

### 3 Background

Reinforcement learning [19] is a particular section of machine learning class that is entirely offbeat from the other two categories (i.e. supervised and unsupervised learning) [20, 21]. The main idea is to keep on learning and producing better results. This is done by firstly monitoring the environment and keep on analysing the situations so as to take suitable action. The reward is earned based on the effectiveness of the actions. So, the goal is to augment the rewards. We can denote the reinforcement learning model with the help of a four-tuple (S, A, R, T) which is described below:

#### A. RFID

The RFID is a special type of technology that employs radio frequency signals to identify, pinpoint, and track the target without human intervention [22, 23]. RFID systems consist of transponders, transceivers, and back-end storage which are commonly called tags, readers, and databases, respectively. Tags have a microchip that has memory constraints and is used to store the unique tag identifier and other related information. They are two types: active and passive. Active tags have their own battery source while passive tags don't. They use destination electromagnetic waves to transmit the collected data. Since these tags have a longer range of transmission thus, they are preferred where there is a need to identify objects over long distances such as roadside units in traffic management, health care applications, animal tracking, object locating in logistics markets, etc. While readers are the devices that can read tags for object identification and stored information. It can transfer these data to the back-end systems.

#### B. Reinforcement Learning Model

For building a traffic light control system using deep Q-learning [17], we need to define the states, actions, and rewards.

Here we present how the three elements are defined in our model.

**State:** The state of the agent describes a representation of the situation of the environment in a given timestep  $t$  and it is denoted with  $s_t$ . For effective learning of agents to optimize the traffic on each road the agent should have sufficient information on the distribution of vehicles in the current environment. The objective of this representation is to allow the agent to know about the environment where vehicles are located at timestep  $t$ . To serve this purpose the approach proposed in this paper is inspired by paper [24] due to its simple representation of state which makes use of RFID reader so easy. In particular, this state design includes only spatial information about the vehicles hosted inside the environment, and the cells used to discretize the continuous environment are not regular. In this paper, we will explore the chance of getting good results from a simple state representation. In each arm of the intersection, incoming lanes will be discretized in cells that will notify whether or not a vehicle is present inside the cell (Fig. 1).



Fig. 1 State representation

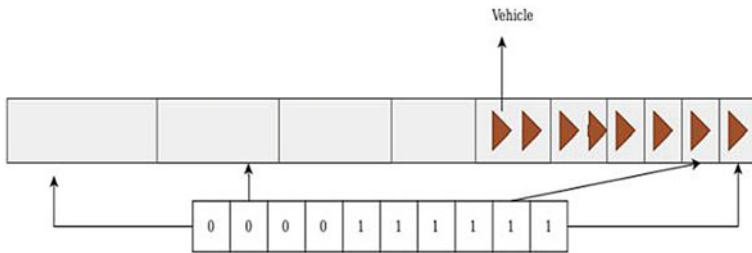


Fig. 2 DSR vector representation with respect to cells of intersection

(1) **Discrete state representation:** Basically DSR (discrete state representation) is a mathematical representation of the state space as a form of the vector where every element is computed by the following equation

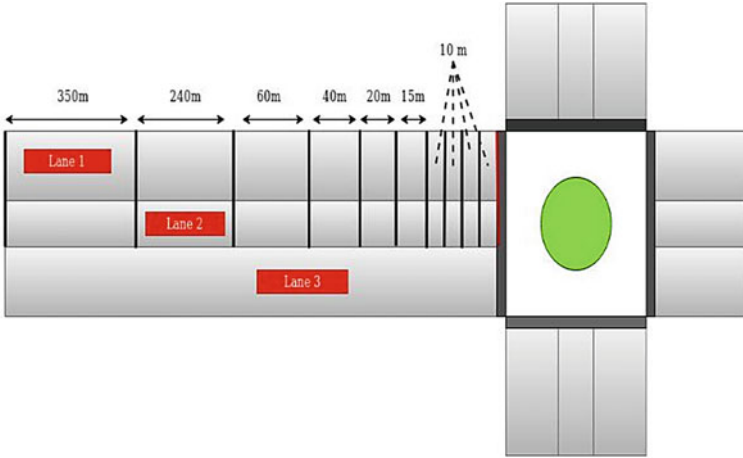
$$DSR[i] = c \tag{1}$$

$c = 1$  if  $i$ th cell contains at least one vehicle else  $c = 0$ , as explained in Fig. 2.

It must be noted that the DSR vector does not represent only a single intersection. As shown in Fig. 1 we have divided a single road of intersection into 3 lanes. As described in Fig. 3 lane1 is dedicated to going straight and left, lane2 is dedicated to going right only. So lane1 and lane2 both are having different DSR vector and each lane contain 10 cells that mean every arm of the intersection there are 20 cells and in the whole intersection 80 cells.

So, the proposed state-space is composed of 80 boolean cells. This means that the number of possible states is  $2^{80}$ . The choice of boolean cells for the environment representation is also crucial because the agent has to explore just the most significant subset of the state space in order to learn the best behaviour.

When the agent samples the environment at a timestep  $t$ , it receives a vector  $DSR_t$  containing the discretize representation of the environment in that timestep. This is the principal information about the environment that the agent receives, so it is



**Fig. 3** State representation of the west road of the intersection, with cells length

designed to be as precise as possible but without being excessively detailed in order to not increase the computational complexity of the neural network’s training [25].

**Action space-**The action set identifies the possible actions that the agent can take. The agent is the traffic light system, so doing an action translates to turning green some traffic lights for a set of lanes and keep it green for a fixed amount of time. In this paper we are using green time is set at 10 s and the yellow time is set at 4 s. In other words, the task of the agent is to initiate a green phase choosing from the predefined ones. The action space is defined as

$$A = \{NSSL; NSR; EWSL; EWR\} \tag{2}$$

The set represents every possible action that the agent can take. Every action a set is described below

- **North-South Straight and Left (NSSL):** the green light phase will be activated for those vehicles which are present in the north and south arm and want to proceed either straight or left
- **North-South Right (NSR):** the green light phase will be activated for those vehicles which are present in the north and south arm and want to proceed with their right arm.
- **East-West Straight and Left (EWSL):** the green light phase will be activated for those vehicles which are present in the east and west arm and want to proceed either straight or left.
- **East-West Right (EWR):** The green light phase will be activated for those vehicles which are present in the east and west arm and want to proceed with their right arm (Fig. 4).

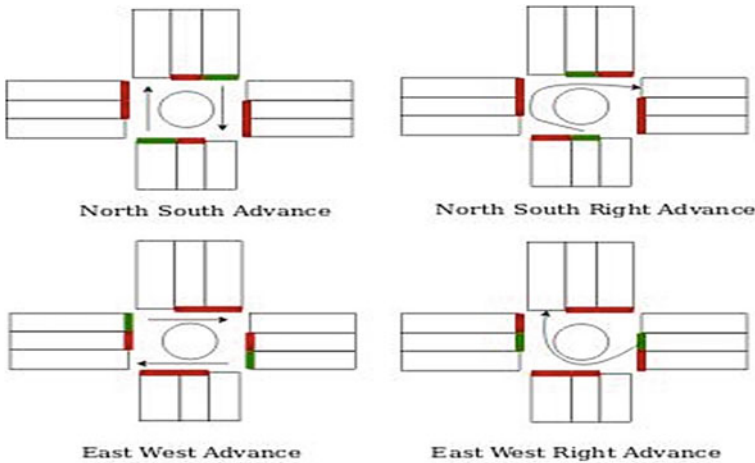


Fig. 4 The four possible actions

If the same is chosen at timestep  $t$  and timestep  $t - 1$  (i.e. the traffic light orientation is the same) then there is no yellow phase and therefore the current green phase continues. And if the action chosen in timestep  $t$  is not equal to the previous action, a 4 s yellow phase is activated between the two actions. As explained in Fig. 5

Rewards: In reinforcement learning, based on the action chosen by the agent—the feedback will be generated from the environment in terms of reward [26, 27]. To enhance the model, based on the reward the agent will change its intuition for further future action. Therefore, the reward is a crucial aspect of the learning process. The reward usually has two possible values: positive or negative. Good action led to positive reward similarly bad action leads to negative reward. In this scenario, the objective is to maximize the traffic flow through the intersection over time. In order to achieve this goal, the reward should be derived from some performance measure of traffic efficiency, so the agent is able to know whether or not the action increases efficiency. In traffic analysis, several measures are used, such as throughput, mean delay, and travel time [28]. In this paper, the agent measures the total waiting time defined as:

Total waiting time: The sum of individual waiting times of each car in the environment in timestep  $t$ . waiting time is defined as the time duration during which a vehicle is moving with a speed of less than 0.1 m/s. The total waiting time is computed by the following equation.

Fig. 5 Possible simulation time Difference between two actions



$$tw_t = \sum wt_{(veh,t)} \tag{3}$$

where  $tw_t$  is the total waiting time at timestep  $t$  and  $wt_{(veh,t)}$  is the time duration (in seconds) a vehicle  $veh$  has a speed of less than  $0.1$  m/s at timestep  $t$ .  $n$  represents the total number of vehicles in the environment in timestep  $t$

Reward function: The reward function that generates a reward for the agent is defined in the equation as

$$r_t = tw_{t-1} - tw_t \tag{4}$$

where  $r_t$  represents the reward at timestep  $t$ .  $tw_t$  and  $tw_{t-1}$  denote the aggregate waiting time of all the vehicles at the intersection at timestep  $t$  and  $t - 1$ , respectively.

- (2) **Positioning of RFID Reader for DSR:** As explained earlier DSR vector is a discrete representation of state from a continuous environment. In the simulation, SUMO provides much functionality to get this vector but for real-time, we proposed the use of RFID reader and RFID tags. Here we are supposing that each vehicle is having an RFID tag and we are using RFID reader at the starting of each cell as shown in Fig. 6. Starting cells are having RFID readers with 2–5 m range capacity and the rest are having with a range of about 10 m capacity. As we explained earlier if there is only one vehicle in the  $k$ th cell then  $DSR[k]$  set to one, so basically here we are trying to detect the vehicle at starting of the cell.

RFID reader for the first 4 small cells: For the first four small cells, the RFID will be positioned at the beginning of the cell facing towards diagonal with the range capacity 5 m as shown in Fig. 7.

RFID reader for other cells: The RFID reader for other cells will be positioned 5 m apart from the beginning of the cell facing towards the road with the range 2 m. Here we are assuming the width of the road is 3.7 5 m as used in India.

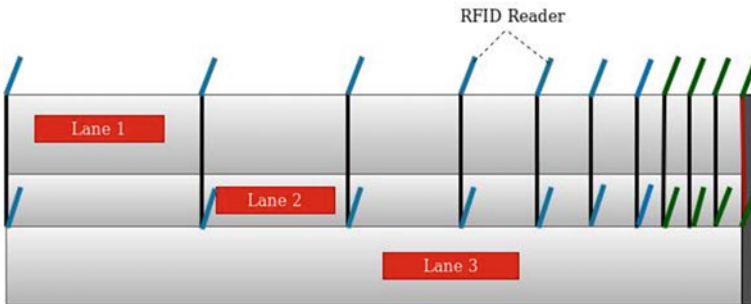
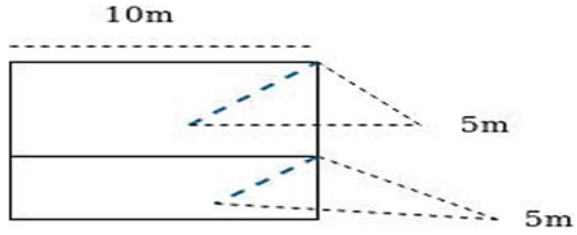


Fig. 6 RFID positions in cells of arm



**Fig. 7** RFID positions for starting 4 cells of the arm



### 4 The Agent’s Learning Mechanism

We have employed Deep Q-learning for the purpose of learning. It is a combination of deep learning and neural networks [25].

#### A. Q-Learning

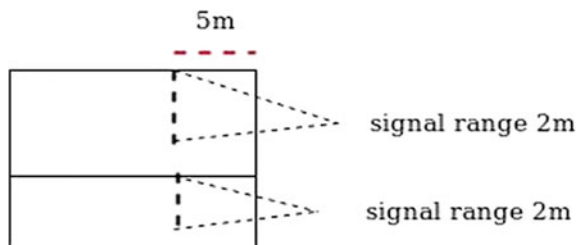
It is a unique form of reinforcement learning which is model-free [29]. According to the state of the environment, a value is assigned to the action which is about to be taken. This value is called the Q-value which is defined in the form of an equation (Fig. 8).

$$Q(s_t, a_t) = Q(s_t, a_t) + (r_{t+1} + \gamma \max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)) \quad (5)$$

Where:

- $Q(s_t, a_t)$  represents the value obtained after the action  $a_t$  has been taken after analysing the state  $s_t$ .
- The above equation basically updates the present Q-value with a quantity discounted by the learning rate.
- $r_{t+1}$  denotes the reward.
- $t + 1$  highlights the relationship between taken action  $a_t$  and the subsequently received reward.
- The immediate future’s Q-value is denoted by  $Q(s_{t+1}, a_t)$ , where  $s_{t+1}$  represents the evolved state after implementing action  $a_t$ .
- Among all the possible actions  $a_t$  the most valuable action is selected which is represented by  $\max A$ .

**Fig. 8** RFID positions for other cells of the arm



- is the discount factor that assumes a value between 0 and 1, lowering the importance of future reward compared to the immediate reward

Here we are going to use a slightly modified version of the equation which is mentioned below:

$$Q(s_t, a_t) = r_{t+1} + \gamma \max_A Q'(s_{t+1}, a_{t+1}) \quad (6)$$

where

- $r_{t+1}$  represent the reward.
- The term  $Q'(s_{t+1}, a_{t+1})$  is the Q-value associated with taking action  $a_{t+1}$  in state  $s_{t+1}$  i.e. the next state after taking action  $a_t$  in state  $s_{t+1}$ .
- As seen in the above equation  $\gamma$  denotes a small penalization of the future reward compared to the immediate reward.

## B. Deep Q-Learning

In order to map a state of the environment  $s_t$  to Q-values representing the values associated with actions  $a_t$ , a deep neural network [25] is built. The input of the network is the vector  $DSR_t$  the state of the environment at timestep  $t$ . The outputs of the network are the Q-values of the possible action from state  $s_t$ .

The input layer of the neural network  $S^{in}$  is defined as:

$$S_{k,t}^{in} = DSR_{k,t} \quad (7)$$

where  $S_{k,t}^{in}$  is the  $k$ th input of the neural network at timestep  $t$  and  $IDR_{k,t}$  is the  $k$ th element of the vector  $DSR$  at timestep  $t$  as shown in Fig. 5 This means that  $|S^{in}| = |DSR| = 80$ , that is the input size of the neural network.

The output layer of the neural network  $S^{out}$  is defined as:

$$S_{j,k}^{out} = Q(s_t; a_{j;t}) \quad (8)$$

where  $S_{j,k}^{out}$  is the  $j$ th output of the neural network at timestep  $t$  and  $Q(s_t, a_{j;t})$  is the Q-value of the  $j$ th action taken from state  $s_t$  at timestep  $t$ . This means that the output cardinality of the neural network is  $|A| = 4$ , where  $A$  is the action space.

The neural network is a fully connected deep neural network with a rectified linear unit activation function (ReLU). And 5 hidden layers are used.

As explained in Fig. 9 shows the vector  $DSR$  as the input of the network, then the network itself with the hidden layers, and finally the output layer with 4 neurons representing the 4 Q-values associated with the 4 possible actions.

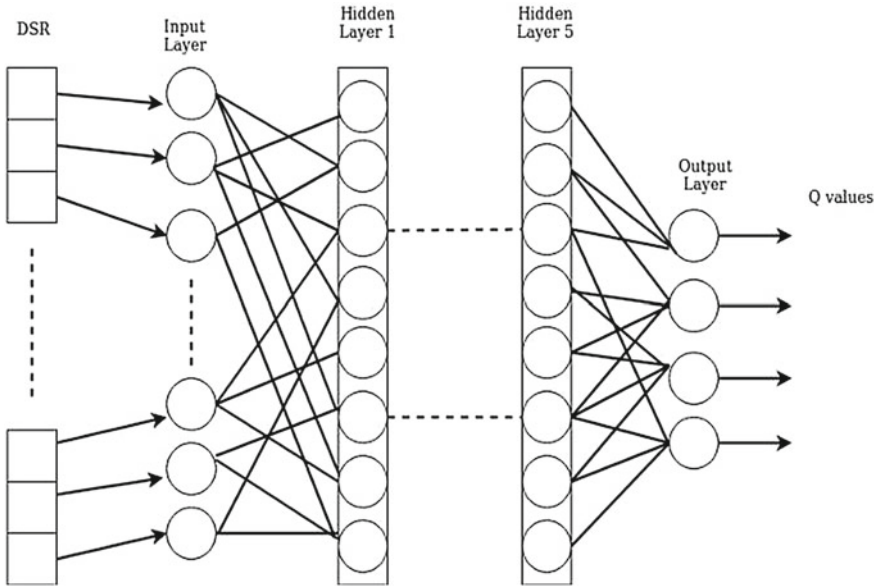


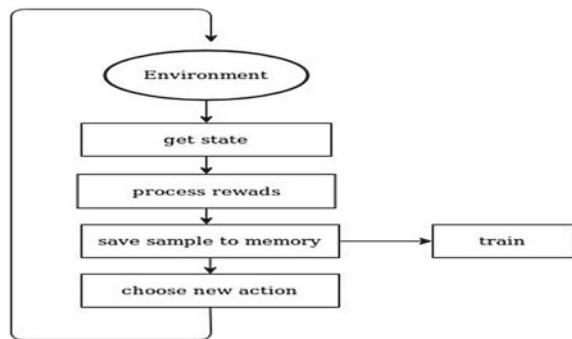
Fig. 9 Strategy of deep neural network

## 5 Experimental Setup and Training

As we have already defined the specification of an agent such as the state, the possible actions, and the reward. Fig. 10 shows how all these components work together to establish the workflow of the agent during one single timestep  $t$ .

After a fixed amount of simulation steps, the timestep  $t$  of the agent begins. First, the agent retrieves the environment state and the delay times next, using delay times of this timestep  $t$  and from the last timestep  $t - 1$ . It calculates the reward associated with action taken at  $t - 1$ . Then the agent packs the information gathered and saves

Fig. 10 The workflow of the agent in a timestep



it to a memory which is used for training purposes. Finally, the agent chooses and set the new action to the environment, and a new sequence of simulation step begins.

### A. Experience Replay

For the sake of improving the conduct of the agent and the learning efficiency, a procedure is endorsed during the training phase which is called Experience replay [30]. It requires acknowledgment of all the essential information necessary for learning to the agent, in the form of a group called a batch. Before submitting simulation information, the agent takes the batch to the data structure intuitively called memory who stores all collected samples. A sample  $m$  is formally defined as the quadruple.

$$m = \{s_t, a_t, s_{t+1}, a_{t+1}\} \quad (9)$$

where  $r_{t+1}$  is the reward that is obtained after carrying out the action  $a_t$  from state  $s_t$ , which unfolds into the succeeding state  $s_{t+1}$ . A training instance involves the gathering of a group of samples from the memory and the neural network training using the aforesaid samples.

### B. The Training Process

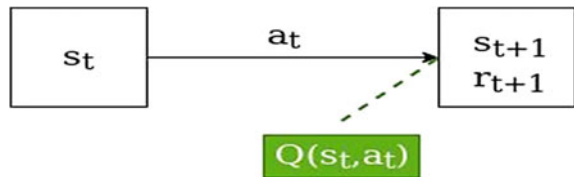
Given the description of experience replay, a detailed explanation of the training process is explained. This process is executed every time a training instance of the agent is initiated.

- A sample  $m$  containing the most recent information, described in Eq. (8), is added to the memory.
- A fixed number of samples (light sampling strategy used) are picked randomly from the memory constituting the batch  $B$ .

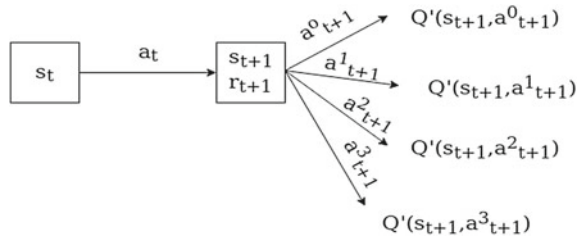
A single sample  $b_k \in B$  contains the initial state  $s_t$  with the most suitable action selected  $a_t$  and its corresponding reward  $r_{t+1}$  following the next immediate state  $s_{t+1}$ . For every sample  $b_k$  the following operations are performed.

- (1) Computation of the Q-value  $Q'(s_{t+1}, a_{t+1})$  by submitting the vector DSR representing  $s_t$  to the neural network and obtaining the predicted Q-value relative to action  $a_t$ . As shown in Fig. 11.
- (2) Computation of Q-values  $Q'(s_{t+1}, a_{t+1})$  by submitting the vector DSR representing the next state  $s_{t+1}$  to the neural network and obtaining the predicted Q-values relative to actions  $a_{t+1}$ . These represent how the environment will evolve and what values will probably have the next actions. As shown in Fig. 12.

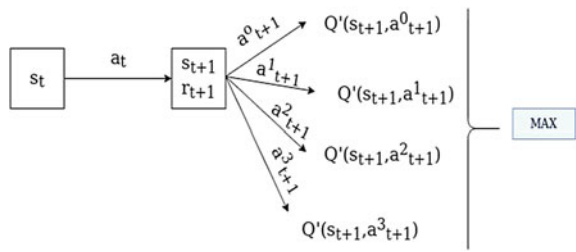
**Fig. 11** Computation of the Q-value  $Q$  for one sample



**Fig. 12** Computation of Q-values  $Q'$  for one sample



**Fig. 13** Update Q-value using Eq. (6) for one sample



- (3) Update of the Q-value using equation (among the possible future Q-values computed in stage two)  $\max_A Q'$  indicates that the best possible Q-value is selected, representing the maximum expected future reward. It will be deducted by a factor  $\gamma$  that gives more importance to the immediate reward. As shown in Fig. 13
- (4) Training neural network: The input is the vector DSR representing the state  $s_t$ , while the desired output is the updated Q-values  $Q(s_t, a_t)$  that now includes the maximum expected future reward due to Eq. (6), the next time the agent encounters the state  $s_t$  or a similar one, the neural network will be likely to output the Q-value of action  $a_t$  that is comprehensive of the best future situation.

### III. Simulation of Urban MObility (SUMO)

The abbreviation SUMO stands for (Simulation of Urban MObility) [31]. It is a traffic microsimulation which provides a software package that allows users to design the road infrastructure and related elements. Among the packages that SUMO offers, in this paper the following were used.

NetEdit was used to design the static elements of the intersection, such as the characteristics of the roads, the distribution of traffic lights, and the lane connections across the intersection.

Package TraCI is used to define the type, characteristics, and generation of vehicles that are going to be in the simulation

### IV. Result

The model is trained with the following hyper-parameters.

Parameters	Values
Neural network	8 layers, 400 neurons each
Memory size	50,000
Episodes	300
RFID range	2 and 5 m

The high gamma means that the agent is aiming at maximizing the expected cumulative reward of multiple consecutive actions. This can be considered as the true RL agent that has a long look ahead and tries to search for the policy that overall gives the best performance with respect to the reward obtained at every step.

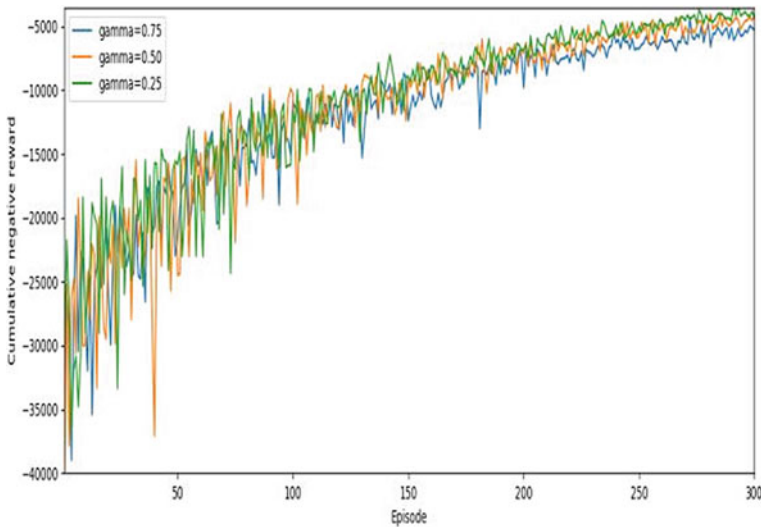
Figure 14 shows the reward gain by RL agent at each episode, as we can see that it continuously increasing which means after each episode agent learns to take better action.

Figure 15 shows the queue length of vehicles i.e. number of cells occupied by vehicles, which is continuously decreasing which means the waiting time of the vehicles is decreasing after each episode.

Figure 16 shows the delay of the vehicle which basically represents the amount of time spent by a vehicle in the red-light phase, as we can see this delay is continuously decreasing after each episode.

### E. Performance Metrics

In order to evaluate the agent performance, after 300 episodes when the agent has been trained, we conduct 10 more episodes to observe the following performance matrix.



**Fig. 14** Reward while training in high-traffic

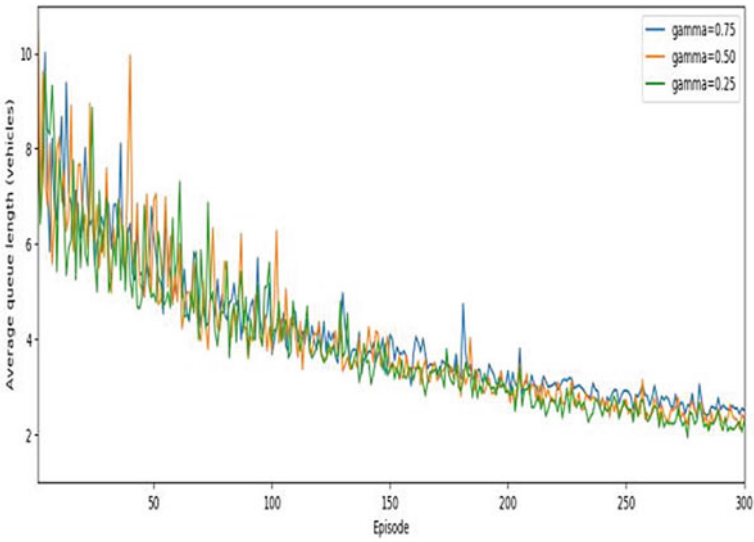


Fig. 15 Queue length of the vehicle while training in high-traffic

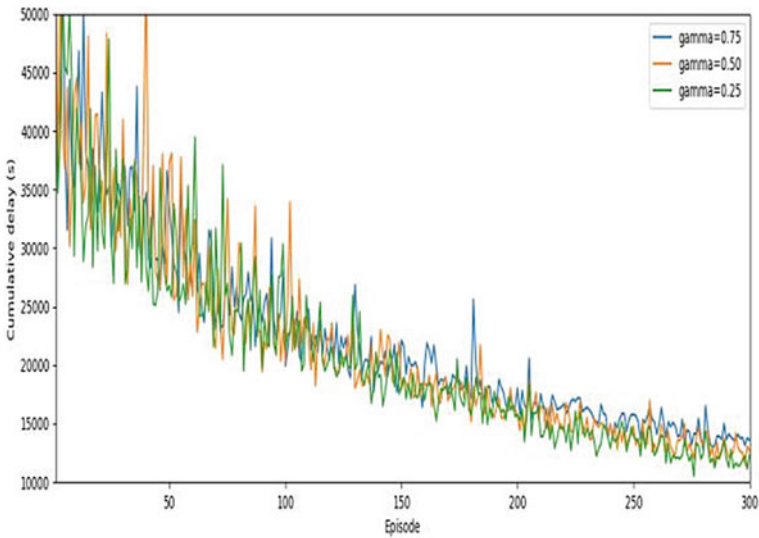


Fig. 16 Delay of the vehicle while training in high-traffic

- (1) Average Negative Reward: Average of all rewards generated by the last 10 episodes when the agent has already trained.

$$anr = avg_{ep} \sum_{ep=0}^{ep=10} r_{ep} \quad (10)$$

(2) Total waiting time: Sum of delay times of last 10 episode.

$$twt = \sum_{ep=0}^{ep=10} wt_{ep} \quad (11)$$

The baseline is needed to compare the agent performance, so we perform a simulation which behaves similarly as the current traffic scenario behave that means every traffic light phase is always activated in the same order defined as

[NSR-NSSL-EWR-EWSL] and following performance matrix observed.

Parameters	Values
anr	-202, 871
twt	942, 652

As Figs. 14 and 16 show that  $\gamma = 0.25$  performs best so that we perform baseline scenario with the same value and in high traffic scenario.

Now the comparison between the baseline scenario and proposed method shown below.

Parameter	Baseline	Proposed	Reduced%
anr	-	-	
	202, 871	44, 155	78.23
twt	942, 652	129, 353	86.27

## 6 Conclusion

Reinforcement learning is actually an environment-dependent algorithm that had achieved augmenting interests in the traffic control domain. In this chapter, we have presented a traffic light control model using a deep Q-learning method (i.e., Q-learning with the neural network) and RFID. The results were discussed. The performance of the proposed model was evaluated using SUMO and was compared. It was shown that the proposed model notably outperforms conventional approaches.

For future work, we would like to investigate other machine learning algorithms for traffic light control systems. Furthermore, we would like to test our model proposed in this study on a more realistic traffic simulator and try to give priorities to important and emergency vehicles like ambulance and a police van.



## References

1. Balaji PG, German X, Srinivasan D (2010) Urban traffic signal control using reinforcement learning agents. *IET Intell Transp Syst* 4(3):177–188
2. Hartenstein H, Laberteaux LP (2008) A tutorial survey on vehicular ad hoc networks. *IEEE Commun Mag* 46(6):164–171
3. Abdulhai B, Pringle R, Karakoulas GJ (2003) Reinforcement learning for true adaptive traffic signal control. *J Transp Eng* 129(3):278–285
4. Qoni LB, Babuska R, De Schutter B (2010) Multi-agent reinforcement learning: An overview. In: *Innovations in multi-agent systems and applications-1*, pp 183–221. Springer
5. Lin L-J, Mitchell TM (1992) Memory approaches to reinforcement learning in non-Markovian domains. Citeseer
6. Lin L-J (1993) Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science
7. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
8. Genders W, Razavi S (2016) Using a deep reinforcement learning agent for traffic signal control. [arXiv:1611.01142](https://arxiv.org/abs/1611.01142)
9. De Schutter B, De Moor B (1998) Optimal traffic light control for a single intersection. *Eur J Control* 4(3):260–276
10. Gao J, Shen Y, Liu J, Ito M, Shiratori N (2017) Adaptive traffic signal control: deep reinforcement learning algorithm with experience replay and target network. [arXiv:1705.02755](https://arxiv.org/abs/1705.02755)
11. Liang X, Du X, Wang G, Han Z (2018) Deep reinforcement learning for traffic light control in vehicular networks. [arXiv:1803.11115](https://arxiv.org/abs/1803.11115)
12. Chu T, Wang J, Codec'a L, Li Z (2019) Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Trans Intell Transp Syst*
13. Khamis MA, Goma W (2014) Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Eng Appl Artif Intell* 29:134–151
14. Li J, Luo G, Cheng N, Yuan Q, Wu Z, Gao S, Liu Z (2018) An end-to-end load balancer based on deep learning for vehicular network traffic control. *IEEE IoT J* 6(1):953–966
15. Gu Y, Lu W, Xu X, Qin L, Shao Z, Zhang H (2019) An improved Bayesian combination model for short-term traffic prediction with deep learning. *IEEE Trans Intell Transp Syst*
16. Ji B, Hong EJ (2019) Deep-learning-based real-time road traffic prediction using long-term evolution access data. *Sensors* 19(23):5327
17. Arel I, Liu C, Urbanik T, Kohls AG (2010) Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intell Transp Syst* 4(2):128–135
18. Mirchandani P, Head L (2001) A real-time traffic signal control system: architecture, algorithms, and analysis. *Transp Res Part C Emerg Technol* 9(6):415–432
19. Sutton RS, Barto AG (2011) Reinforcement learning: an introduction, 2nd edn. <http://incompleteideas.net/sutton/book/the-book-2nd.html>
20. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Hassabis D et al (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533
21. Wiering MA (1999) Explorations in efficient reinforcement learning, Doctoral dissertation, University of Amsterdam
22. Juels A (2006) Rfid security and privacy: a research survey. *IEEE J Sel Areas Commun* 24(2):381–394
23. Want R (2006) An introduction to RFID technology. *IEEE Pervasive Comput* 5(1):25–33
24. Vidali A, Crociani L, Vizzari G, Bandini S (2019) A deep reinforcement learning approach to adaptive traffic lights management. In: *WOA*, pp 42–50
25. Srinivasan D, Choy MC, Cheu RL (2006) Neural networks for real-time traffic signal control. *IEEE Trans Intell Transp Syst* 7(3):261–272
26. Watkins CJCH, Dayan P (1992) Q-learning. *Mach Learn* 8(3–4):279–292
27. Watkins CJCH (1989) Learning from delayed rewards

28. Dowling R (2007) Traffic analysis toolbox volume vi: definition, interpretation, and calculation of traffic analysis tools measures of effectiveness. Technical report
29. Hausknecht M, Stone P (2015) Deep recurrent q-learning for partially observable mdps. In: 2015 AAAI fall symposium series
30. Lin L-J (1992) Reinforcement learning for robots using neural networks
31. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of sumo-simulation of urban mobility. *Int J Adv Syst Measur* 5(3 &4)