Volume 14 - Issue 6 (2025)

Journal of Transportation Science and Technology

Journal homepage: https://tapchicongnghegtvt.vn

Utilizing Reinforcement Learning in Traffic Light Management at an Isolated Intersection

Hung Tuan Trinh[1,*], Van Bac Nguyen[2] , Duy Quang Tran[3]

[1]*Institute of Transport Economics and Development, Ho Chi Minh University of Transport*
[2]*Faculty of Civil Engineering, VNU Hanoi, University of Engineering and Technology*
[3]*Faculty of Civil Engineering, Nha Trang University*

Keywords:

Reinforcement learning
Double Deep Q network
Heterogeneous traffic conditions
Simulation
Motorcycle

ABSTRACT

The rapid pace of urbanization and increasing travel demand have placed significant strain on transportation systems, particularly in densely populated urban areas. A key challenge is congestion at intersections, where high vehicle volumes often lead to long queues, extended travel times, increased environmental pollution, and economic inefficiencies. Traditional fixed-time signal control (FTSC) is insufficient to address these dynamic traffic patterns. In response, Reinforcement Learning (RL) has emerged as a promising approach for adaptive traffic signal control. This paper presents a Double Deep Q-Network (DDQN)-based model for optimizing traffic light control at a single urban intersection. DDQN is a significant advancement in RL by combining the function approximation capability of deep neural networks (DNNs) with the Double Q-learning framework, effectively reducing overestimation bias and improving the stability of value-based learning. This integration enables RL to scale to complex real-world tasks such as traffic light control, where the state space is too large to represent explicitly with a table. The proposed DDQN model enhances traffic performance by dynamically adjusting signal phases under heterogeneous traffic conditions. Experimental results demonstrate the potential of the DDQN approach to enhance traffic speed and reduce congestion compared to conventional strategies.

1. Introduction

Rapid urbanization has caused many negative impacts on traffic conditions in large cities, including traffic jams, infrastructure overload, and environmental pollution. Traffic congestion occurs when travel demand exceeds the capacity of the road infrastructure. The problem of congestion not only affects travel but also causes great damage to the economy. According to statistics, by the end of 2023, Ho Chi Minh City had about 10 million vehicles, including more than 7.6 million motorbikes and 700,000 cars [1]. The rapid increase in personal vehicles exceeds the capacity of the current traffic infrastructure, causing serious congestion. In addition, it also increases fuel costs, reduces

* Hung Tuan Trinh. Institute of Transport Economics and Development, Ho Chi Minh University of Transport
Email: hungtt@ut.edu.vn
https://www.doi.org/10.55228/JTST140609
Received: September 28, 2025; Received in revised: November 13, 2025; Accepted: November 15, 2025
Available online: November 15, 2025
pISSN: 1859-4263; eISSN: 3030-4261

economic productivity, and causes great damage to businesses. Traffic congestion also contributes to increased environmental pollution. When vehicles move slowly or stop, the amount of toxic emissions released into the environment increases. According to statistics, traffic pollution accounts for about 70% of total emissions in Ho Chi Minh City [2].

To reduce traffic congestion in urban areas, many studies and solutions have been proposed. Road expansion is a common method when it is necessary to reduce congestion and improve the ability of vehicles to pass through. However, this solution also has limitations, such as high land clearance costs, and there are areas where land clearance is difficult, such as the core urban area. Therefore, new solutions are needed to control traffic lights more effectively while still maintaining the existing road network infrastructure.

Controlling traffic lights effectively is a broad research topic and is approached in many different directions. Author Webster proposed formulas for calculating the length of the light phase for a single intersection, assuming that the traffic flow is a constant flow in a fixed time, and calculating the optimal light cycle and phase to minimize waiting time [3]. Due to the lack of real-time data, the current popular traffic lights in Vietnam are still controlled by this fixed-phase method. Accordingly, the transition of the light phases follows a sequential order, and the time of the light phases is also set in advance.

Another approach to solving the problem of congestion at intersections is to change the fixed-phase control method to better match the changes in traffic flow. The fixed-phase control method has a constant cycle, leading to an unreasonable situation when traffic flow changes. For example, one direction with high traffic density has to wait for the red light, while the other direction has no traffic at the green light, wasting time and increasing congestion. The Max-Pressure method prevents the intersection from being oversaturated by optimizing the number of approaching and exiting vehicles in each direction [4]. Some other algorithms are also applied to be able to control the signal lights better, such as the Sydney coordinated adaptive traffic system (SCATS) [5], the split-cycle offset optimization technique (SCOOT) [6], or adaptive control software lite (ASC-Lite) [7]. According to these studies, smart signal lights that automatically adjust according to actual traffic will help regulate traffic more flexibly. Accordingly, the time of each signal phase will be adjusted automatically based on the number of vehicles approaching the intersection. However, most of these methods consider the same effects for all modes, meaning that each mode in the traffic flow is treated similarly. In addition, these algorithms do not take full advantage of modern traffic data sources because traditional devices often only detect the presence of vehicles when passing. Therefore, they are difficult to apply to complex intersections. In recent times, emerging technology techniques in the field of automobiles and communication have significantly improved the efficiency of automobiles. Connected Vehicles (CVs) are considered a new solution to enhance road capacity, increase traffic speed, and reduce congestion on the road [7]. Accordingly, CVs are capable of sharing data directly with each other or with infrastructure devices through a wireless communication system [8].

With the development of science and technology, data collection from vehicles on the road can be done through wireless communication systems. The data collected during the movement of the vehicle includes location, speed, waiting line length, and waiting time at the intersection when encountering a red light. This real-time data can be used as input for the Reinforcement Learning (RL) model to control the traffic lights.

In the past, due to limited technical conditions, the Q-learning model [9] often used small-sized state functions and linear functions to calculate the approximate value of Q. However, the complexity of the model has not been fully demonstrated due to limited input data. Additionally, it exhibits several limitations that can impede its effectiveness in complex scenarios. One significant drawback is its reliance on a discrete Q-table to store state-action values, which becomes impractical as the state-action space grows exponentially. This issue, known as the "curse of dimensionality," can lead to inefficient learning and storage challenges in large-scale problems. Additionally, Q-learning can be slow to converge, requiring extensive exploration to accurately estimate Q-values, which is time-

consuming and computationally expensive. The algorithm also struggles with generalization, as it learns specific Q-values for each state-action pair without leveraging similarities between states, limiting its ability to handle unseen situations effectively. Moreover, Q-learning assumes a stationary environment and may not adapt well to dynamic changes, leading to suboptimal performance in real-world applications. These challenges have prompted the development of advanced techniques like Deep Q-Networks (DQN) [10], which utilize neural networks to approximate Q-values and address some of these limitations.

With the development of algorithms, the Deep Neural Network (DNN) has been used to take advantage of the large amount of input data in the DRL model. However, the standard DQN framework suffers from several limitations, most notably overestimation bias, which arises from using the same Q-network to both select and evaluate actions. To address this limitation, Double Deep Q-Network (DDQN) has been introduced as an extension of DQN, which decouples action selection from action evaluation to mitigate overestimation bias and stabilize the learning process [11].

In addition, most DRL models for traffic signal control rely on the assumption that vehicles are equipped with wireless communication systems for continuous data collection. In practice, this assumption is unrealistic, as only a subset of vehicles are equipped with such technologies. Furthermore, the majority of existing studies have been conducted under simulation settings where cars are the dominant mode of transport. This creates a substantial gap when applying these approaches to the Vietnamese context, where traffic flow is predominantly characterized by motorcycles, leading to significant differences.

The main contributions of this paper are shown as follows:

• We employ the DDQN algorithm to optimize traffic signal control under heterogeneous traffic conditions. By decoupling action selection from action evaluation, DDQN effectively mitigates the overestimation problem inherent in standard DQN, thereby improving learning stability and control performance. The proposed model is evaluated against conventional traffic signal control methods, and the results demonstrate the superior effectiveness of employing DDQN for traffic management at an isolated intersection;

• We set up adjustment parameters to simulate the flow of motorcycles in mixed traffic conditions in Vietnam. Motorcycles have different characteristics while moving compared to cars, such as lane sharing, side-by-side riding, and frequent overtaking within a single lane.

The remaining sections of the paper are organized as follows: Section 2 presents the research methodology. Section 3 describes the experimental setup. Section 4 discusses the simulation results and evaluation. The final section provides discussions and conclusions.

2. Research Methodology

This section presents a method for controlling traffic lights at a single intersection based on the application of the DDQN algorithm. Section 2.1 gives an overview of RL, and Section 2.2 presents the principle of the DDQN algorithm.

2.1. Reinforcement Learning

Reinforcement learning [12] is a machine learning (ML) technique to train agents to make decisions to achieve the most optimal results. This technique simulates the trial-and-error learning process that humans use to achieve a goal based on a pre-determined reward function.

In RL, agents are trained to make decisions in a given environment, aiming to maximize the cumulative reward. The agent learns by interacting directly with the environment. Each action of the agent will create a new state and a corresponding reward. This process is repeated continuously, allowing the agent to gain experience and adjust its behavior. The goal of the agent is to maximize the total cumulative reward during the interaction with the environment.

RL is based on the mathematical foundations of the Markov Decision Process (MDP) [12], which provides a formal framework for modeling decision-making in environments where outcomes are partly

random and partly under the control of an agent. An MDP is formally defined as a tuple (S, A, P, R, γ).

At time t, the agent is in state $s \in S$ (S is the state space). It performs an action $a \in A$ (A is the action space) based on the policy π and receives a reward r, and the environment also changes from state s to s'. The probability of transitioning from state s to s' after performing action a is P(s'|s,a). γ is a discount factor that determines the importance of the future reward. α is the learning rate used to control the rate of value update. Q-learning [9] is an algorithm designed to solve problems modeled as MDPs. The Bellman equation [13] forms the foundational basis for Q-learning, a value-based reinforcement learning algorithm aimed at finding an optimal policy by estimating the action-value function $Q^{\pi}(s,a)$. It expresses the recursive relationship between the value of a state-action pair and the expected return from the subsequent state, incorporating both immediate rewards and future returns. Mathematically, the Bellman optimality equation for the Q-function is defined as

$$Q^{\pi}(s,a) = \mathrm{E}\left[r + \gamma \max_{a'} Q^{\pi}(s',a') \mid s,a \right] \qquad (1)$$

where $Q^{\pi}(s,a)$ is the expected return of taking action a in state s under policy π . s' and a' are the next state and action of the agent.

2.2. Double Deep Q Network

Q-learning [9] is a foundational RL algorithm that has demonstrated success in various applications. However, as the dimensionality of the input space increases, the performance of Q-learning deteriorates significantly. DQN [10] uses DNN instead of Q-tables to estimate Q values, which helps to handle large state spaces. The DQN algorithm is a breakthrough in the field of RL, allowing agents to learn to make decisions in complex environments by combining Q-learning with DNN. In DQN, a DNN is employed to approximate the Q-value, which represents the expected future reward for a given action in a particular state. Unlike traditional Q-learning, where the Q-table is directly updated with each action, DQN updates the network itself by minimizing the loss function, which measures the discrepancy between the predicted Q-values and the target Q-values. However, it suffers from overestimation because the same Q-network is used for both action selection and evaluation. DDQN [11] addresses this issue by decoupling these processes, using the online network for action selection and the target network for evaluation.

At state s_t , the agent (traffic signal light) selects an action (phases in action space) a_t to execute, and the environment responds with an immediate reward r_t and transitions to the next state s_{t+1} .This transition tuple (s_t, a_t, r_t, s_{t+1}) is stored in an experience replay buffer, which enables the agent to decorrelate consecutive samples and reuse past experiences. During training, a mini-batch of transitions is sampled from the replay memory to update the network.

In the DDQN, two sets of parameters are maintained to mitigate the overestimation bias inherent in standard DQN: the online network with weights (θ) and the target network with weights (θ^{-}). The online network is responsible for selecting actions, while the target network is used to evaluate them. For a given transition (s, a, r, s') the DDQN target is defined as

$$y^{DDQN} = r + \gamma Q\left(s', \arg\max_{a'} Q(s',a';\theta); \theta^{-} \right) \qquad (2)$$

where γ is the discount factor. The learning objective is then to minimize the mean squared error between the predicted Q-values and the DDQN target, expressed as

$$\mathrm{L}(\theta) = \mathrm{E}\left[\left(y^{DDQN} - Q(s,a;\theta) \right)^2 \right] \qquad (3)$$

This formulation stabilizes the learning process and reduces the overestimation bias, thereby enhancing the robustness of value-based reinforcement learning. In DDQN, the update procedure distinguishes between the online network and the target network. During training, the online network (θ) is updated iteratively through stochastic gradient descent to minimize the temporal difference (TD) loss. Gradient descent is applied to

update the online network with the learning rate (α) as below:

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathrm{L}(\theta) \quad (4)$$

Conversely, the target network (θ^{-}) is updated less frequently by synchronizing it with the online network parameters ($\theta^{-} \leftarrow \theta$) after a fixed number of steps, constituting a hard update strategy. The flowchart of DDQN is expressed in Figure 1.

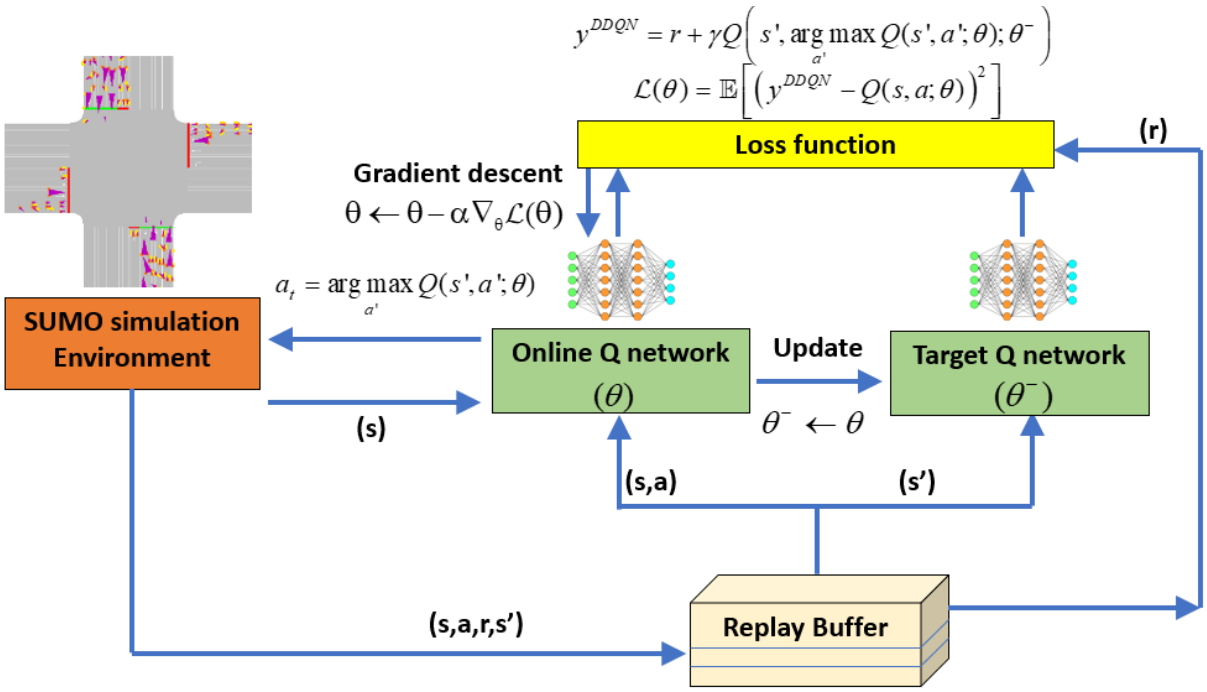

Figure 1. Flowchart of DDQN.

3. Experimental Setup

In this paper, we use the Simulation of Urban Mobility (SUMO) [14] tool to simulate the operation of traffic lights, shown in Figure 2. The intersection can be accessed from 4 directions, each direction will have 4 lanes and be divided into 1 separate left turn lane and 2 straight lanes, 1 shared lane between right turn and straight.

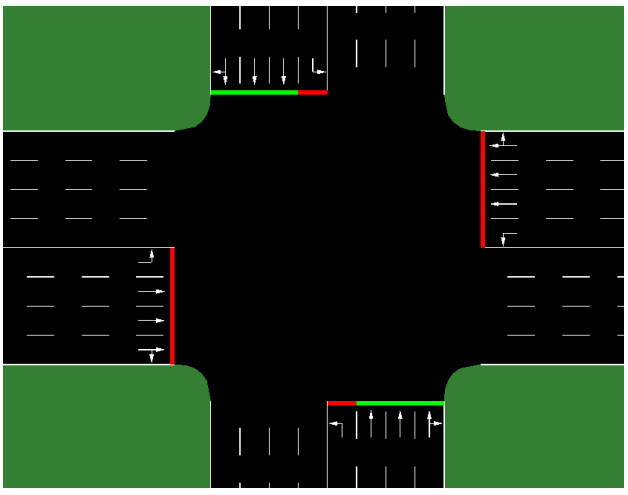

Figure 2. Intersection Plan and directions.

Traffic volume is at a moderate level, including both cars and motorbikes. The model uses data collected from vehicles to control traffic lights. Traffic volume in the model is randomly generated according to the pre-defined probability in the .rou file.

The DDQN model is combined using the Python programming language and the Tensorflow library when training. The SUMO Traci protocol uses a server/client architecture to provide a way to access SUMO and extract data from the simulation model.

3.1. Proposed DDQN model to control traffic light

- *State Space*

The state of the signal agent is used to describe the environment at time t. This state needs to provide enough information on the approaches so that the agent can learn and choose the optimal action, that is, the signal can choose the best phase sequence and phase time.

The model uses the Discrete Traffic State Encoding (DTSE) technique [15] to represent the state of the signal agent based on the presence of vehicles. The approaches to the intersection will be divided into small cells from the stop line to a distance of 800m. The choice of the length of these cells directly affects the operation as well as the efficiency of the signal light. Cells located closer to the intersection are assigned shorter lengths to provide higher resolution data where vehicle behavior is most critical for phase decision-making. This structure ensures that detailed traffic dynamics near the junction are accurately captured. Each cell is dimensioned to accommodate at least a single vehicle, enabling a representation that simplifies the state space while preserving essential traffic flow characteristics.

From the above analysis, the length of these cells at a typical approach is shown in Figure 3 below. Each approach to the intersection is discretized into 24 cells, comprising 12 cells for the dedicated left-turn lane and 12 cells collectively representing the remaining three through and right-turn lanes. This configuration reflects the signal phase operation, where the left-turn movement is controlled by a separate phase, while the other three lanes operate under a shared phase. As the intersection consists of four approaches, the complete state representation includes 96 cells.

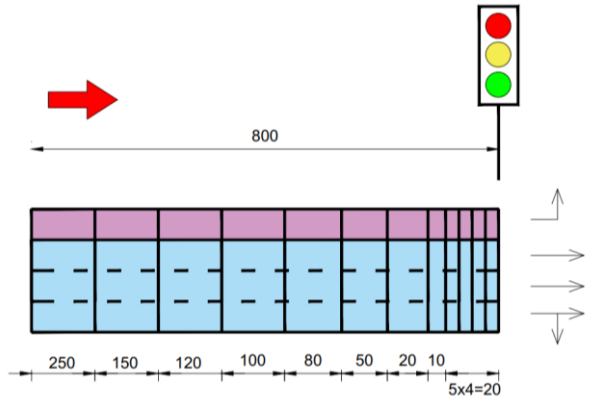

Figure 3. Data collection at an approach intersection.

The state vector is defined as

$$S_t = [v_0, v_1, ..., v_{95}] \quad (5)$$

Where v_i is the average normalized speed of vehicles in this cell, and it is calculated as

$$v_k = \frac{mean \text{ speed of vehicles in cell k}}{v_{\max}} \quad (6)$$

Where $v_{\max}$ is the maximum speed of vehicles. It is a continuous state representation where each element lies in [0,1], capturing relative mobility levels within each discretized road segment. Cells without vehicles are assigned a value of zero, indicating the absence of traffic in those locations.

- Action Space

The typical cycle of a traffic light includes green phase, yellow phase, and red phase. In our model, the action space is determined based on the activity of the traffic light phase, that is, the duration of the green phases. In the DNN model, after each duration (Δ_t =5s), the agent will choose an action to perform, that is, choose one phase from the four phases in Figure 4. If the selected action is the same as the one taken in the previous time step, the current green phase is extended. However, if the action differs from the previous one, the green phase transitions to a yellow phase, and the signal moves to the next green phase. The duration of each green phase may vary, as it is determined by the agent's state at each time step.

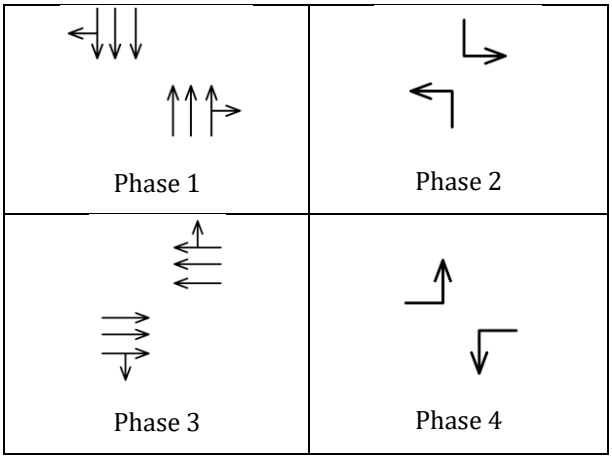

Figure 4. Traffic signal phases.

- Reward Function

The reward function is an important part of RL because it determines the goal of the model after training. In our model, the reward function was designed to reduce traffic congestion and improve flow performance. Due to the influence of traffic lights and conflicts, the vehicles have to slow down or stop. By penalizing the number of halted vehicles and

accumulated waiting time at an intersection, the reward function aims to minimize stops and improve overall throughput. To encourage efficient traffic flow, the mean speed across all incoming roads was incorporated as a positive reward signal. Formally, the reward function is defined as:

$$\text{Reward} = -\text{Number of Halting vehicles} - \text{waiting\_time}*0.05 + \text{mean\_speed}*0.1 \quad (7)$$

Since the number of halting vehicles, waiting time, and vehicle speed are measured in different units and vary significantly in magnitude and direct aggregation would result in one factor dominating the reward signal. Scale parameters (0.05 with waiting time and 0.1 with mean speed) were incorporated into the reward function to convert heterogeneous traffic performance indicators into a comparable scale.

- *Hyperparameters in DDQN*

An activation function helps the network learn complex patterns by introducing non-linearity. In artificial neural networks, the activation function plays an important role in determining the output of a node based on its input or set of inputs. We use the RELU function [16] in the model because of its simplicity and non-linearity, which can learn complex data patterns.

The model uses the popular Adam optimization algorithm [17] to train the DNN. Adam is an adaptive learning algorithm designed to improve the training speed in DNN and achieve rapid convergence. This algorithm is particularly suitable for training DNN because it calculates individual adaptive learning rates for different parameters. Adam combines Momentum and RMSprop techniques to provide a more balanced and efficient optimization process.

Experience replay is a crucial component in DDQN that enhances learning stability and efficiency. Instead of learning from consecutive experiences, which are often highly correlated, experience replay stores the agent's experiences, each consisting of a state, action, reward, next state, and done flag, in a replay buffer. During training, the DDQN randomly samples mini-batches from this buffer to break temporal correlations and smooth out learning updates. This approach allows the network to learn from a more diverse set of experiences, leading to better generalization and more stable convergence. Additionally, experience replay enables the agent to reuse past experiences multiple times, improving data efficiency and making learning more robust, especially in complex environments.

In the DDQN algorithm, epsilon (ε) is an important parameter that controls the balance between exploration and exploitation. Exploration allows the agent to discover new actions that may lead to larger rewards in the future, while exploitation allows the agent to use learned knowledge to maximize immediate rewards. DDQN uses an epsilon-greedy strategy for action selection. Under this strategy, with probability epsilon, the agent chooses a random action from the action space. Conversely, with probability 1 - epsilon, the agent chooses the action with the highest estimated Q value (the "greedy" action). The formula to determine ε is defined below

$$\varepsilon = \varepsilon_{end} + (\varepsilon_{start} - \varepsilon_{end}) e^{-episode/\varepsilon_{decay}} \quad (8)$$

At the initial time (epsilon = 1), the agent explores more, trying out different actions to gather more information. As epsilon decreases, the agent focuses more on exploiting the best-known actions, leveraging its experience.

In DNN, the number of layers and the number of neurons per layer are crucial hyperparameters that directly influence the model's capacity to learn complex patterns. These parameters are used to control the learning process of the DNN algorithm. They are determined before the training process begins. Choosing too many layers and too many neurons in each layer can cause over-fitting and increase the training time. On the contrary, choosing too few models can cause the model not to choose the appropriate action (under-fitting). In this study, we employ a two-layer feedforward network with 128 neurons in each layer. This architecture offers adequate representational capacity to process the 96-dimensional state input while maintaining stable convergence throughout training, as illustrated in Figure 5.

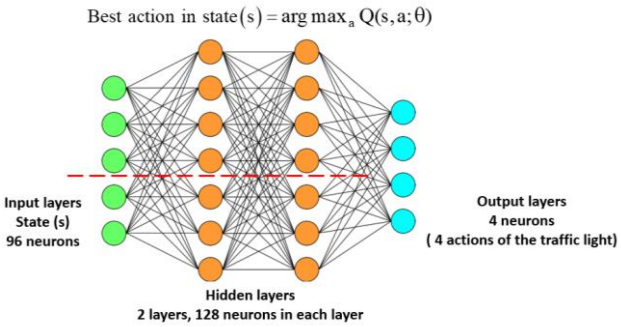

Figure 5. Neural network with 2 hidden layers.

The models were trained 200 times, each time lasting 1200 seconds. Table 1 below summarizes the hyperparameters of the DDQN model.

Table 1. Training parameters of DDQN.

Parameter	Value
γ (Gama)	0.99
α (learning rate)	0.001
Number of states	96
Number of actions	4
Number of layers	2
Number of neurons in each layer	128
$L(\theta)$ (Loss Function)	Mean Squared Error
Activation Function	Relu
Optimization Function	Adam
Total Episodes	200
Step in each Episode	1200s
Batch size	64
Memory size (min-max)	(1000-50.000)
Δ_t (Green duration)	5s
Yellow duration	5s
Epsilon_start	1
Epsilon_end	0.01
Epsilon_decay	15

3.2. Car and Motorcycle Models

The behavior of individual vehicles is governed by a set of microscopic traffic models, including car-following, lane-changing, and junction models. The car-following model, most commonly implemented as the Krauss model [18], is a fundamental component in traffic simulation used to replicate driver behavior in longitudinal vehicle movement. This model captures how a driver continuously adjusts their vehicle's speed in response to the distance to the vehicle in front, striving to maintain a safe headway while also promoting smooth and efficient traffic flow. By incorporating factors such as acceleration, deceleration, desired speed, and randomization to reflect human behavior variability, the Krauss model effectively balances safety with traffic efficiency. Its widespread adoption in microscopic traffic simulators makes it a reliable choice for studying vehicle interactions, congestion dynamics, and evaluating traffic control strategies.

Lane-changing behavior is modeled using rule-based approaches such as LC2013 [19], which account for both mandatory (e.g., to reach an exit) and discretionary (e.g., to achieve higher speeds) lane changes, considering factors like surrounding vehicle speeds and safety constraints. At junctions, SUMO uses a heuristic-based junction model that handles right-of-way rules, traffic signals, and priority settings to determine vehicle movement through intersections. Together, these models enable detailed and realistic simulation of complex traffic scenarios in urban networks.

Although SUMO offers many default models for cars, it does not provide a separate model for motorcycles. Motorcycles are simulated by using default car models, which results in an incomplete and inaccurate representation of real motorcycle behavior. In Vietnam, the traffic composition is dominated by motorcycles, and their behavior diverges significantly from standard traffic models. Motorcycles often travel side-by-side (in parallel) within a single lane, as shown in Figure 6, exploiting the entire lane width much more efficiently than cars. While cars generally follow strict lane discipline with typically one car per lane, dozens of motorcycles can occupy the same space, often ignoring formal lane markings, especially in dense urban settings or during rush hour. This leads to a high-density, flexible flow that is not well-captured by conventional lane-based car-following and lane-changing models.

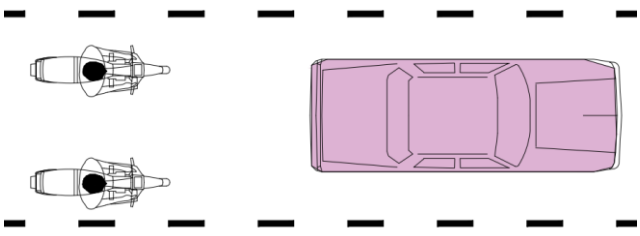

Figure 6. Movements of cars and motorcycles in mixed traffic conditions.

We use two vehicle types, "motorcycle" and "passenger car", in the ".rou" file, to simulate the operation of mixed traffic conditions in Vietnam. The parameters used to simulate motorcycles and passenger cars are shown in Table 2 below.

Table 2. Car and Motorcycle models in simulation.

Parameters	Car	Motorcyle
Length (m)	5	1.6
minGap (m)	2.5	1
maxSpeed (m/s)	20	20
Sigma (driver imperfection parameter)	0.8	0.5
Tau (desired minimum time headway).	1	0.5

In Sumo, the two parameters sigma and tau directly affect the movement behavior of vehicles on the road. The sigma coefficient determines the level of randomness in driving behavior; it affects traffic speed and lane changes. Tau is the reaction time when there is a change in traffic. It affects the braking ability and the safe distance between vehicles.

To simulate the behavior of motorcycles in traffic, we use some additional parameters as expressed in Table 3. Accordingly, motorcycles exhibit lateral flexibility within a lane, allowing them to pass each other within the same lane. They can weave through gaps, ride two or more abreast, and dynamically adjust their position across the width of the lane. This behavior is non-lane-based or sub-lane-based and contrasts sharply with cars, which follow stricter lane discipline.

Table 3. Sublane model for motorcycles.

Parameters in the lane changing model		Value
lcStrategic (The eagerness for performing strategic lane changing)	[0-infinity)	10
lcSpeedGain (The eagerness for performing lane changing to gain speed)	[0-infinity)	5
lcKeepRight (The eagerness to follow the obligation to keep right)	[0-infinity)	5
Lateral-resolution (*)		1.5

This feature (*) enhances the realism of vehicle movement within a traffic simulation. Unlike the default lane-based model, where vehicles are confined strictly to the center of each lane, the sublane model enables more flexible lateral movements, allowing multiple vehicles to occupy slightly offset positions within the same lane. The width of a vehicle relative to the lateral resolution (lateral-resolution) or sublane width determines how many sublanes it occupies or needs to occupy [20]. This capability is particularly useful for simulating realistic behaviors such as overtaking a slower motorcycle on a single lane without requiring a full lane change. Additionally, in high-density traffic conditions, the sublane model allows for the formation of virtual lanes, where up to three motorcycles can travel side by side on two adjacent lanes, reflecting real-world phenomena like compact vehicle formations at intersections or during traffic jams. In our model, both motorbikes and cars can still be in one lane while waiting at the stop line.

4. Simulation Results and Evaluation

To evaluate the effectiveness of the proposed DDQN-based traffic signal control algorithm, five simulation scenarios were designed for comparative analysis.

- DDQN: this method uses two separate neural networks—an online network and a target network—to reduce overestimation bias during action-value updates. The online network selects the optimal action based on current state estimates, while the target network evaluates the value of that action, thus decoupling action selection and evaluation.

- DQN: Unlike DDQN, the standard DQN employs a single network for both action selection and evaluation, which can result in biased Q-value estimates and unstable learning.

- Q-learning is a classical tabular RL method that iteratively updates value functions to determine effective phase selection without requiring function approximation. The speed value of the first cell in the state vector is converted into a binary variable (1 if normalized speed ≥ 0.5, otherwise 0). This design significantly reduces the dimensionality of the state

space while still preserving the essential information for learning.

- Actuated Traffic signal control (ATSC): is a method that operates through sensor-based detection, where green phases are extended or terminated depending on real-time vehicle arrivals at intersections. Detectors are placed on incoming lanes near the stop line. Each green phase begins with a minimum green time to ensure safe vehicle clearance. If no vehicle arrives within the detector gap, the green phase terminates. If vehicles are still present, the green is extended but only up to the max-gap threshold.

- FTSC: In this method, the duration of green phases and the sequence of signal phases remain constant throughout the simulation. This approach serves as a baseline, reflecting traditional, non-adaptive traffic signal operation.

By contrasting these four setups, we aim to assess the ability of the learning-based controller to improve traffic flow efficiency under heterogeneous traffic, including cars and motorcycles.

The experiments were conducted on a system with an Intel Core i9-10900 CPU at 2.80 GHz and 32 GB of DDR4 RAM at 2666 MHz. An NVIDIA GeForce RTX 3070 GPU with 8 GB VRAM was used to accelerate parallel computation and deep learning tasks.

4.1 Model efficiency based on reward function

We evaluate four models' performance using the reward function and other metrics. Throughout the training process, as shown in Figure 7, the reward curves of DQN and Q-learning exhibit an overall upward trend, indicating an improvement of the RL models. Although the reward curve displays fluctuations, particularly during the early stages of training, these variations diminish over time. This behavior is explained by the agent's exploration-exploitation strategy. At the beginning($\varepsilon = 1$), the agent predominantly explores the environment by selecting random actions (e.g., random phase sequences), resulting in unstable performance. As ε decreases, the agent shifts towards exploitation, improving decision-making.

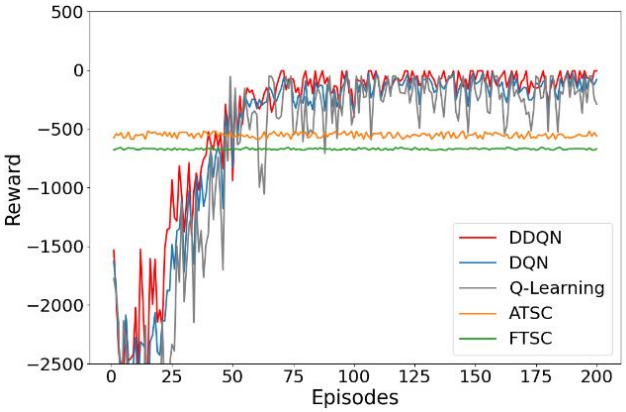

Figure 7. Reward curve of algorithms over 200 episodes.

In Figure 7, the DDQN achieves the highest and most stable reward after around 50 episodes, indicating that it learns an optimal policy faster and converges more reliably than the other methods. The use of a separate target network to mitigate overestimation bias allows DDQN to outperform DQN and Q-learning in both stability and peak reward. The DQN also shows strong performance, with a clear upward learning curve and convergence toward high rewards. However, compared to DDQN, DQN exhibits more oscillations and a slower convergence rate due to overestimation issues inherent in the update process. Q-learning also improves significantly after initial fluctuations, though its convergence is less stable than DQN, reflecting the limitations of tabular representation and discretization in capturing complex traffic dynamics. In contrast, ATSC and FTSC maintain relatively stable but consistently lower rewards throughout all episodes. ATSC performs slightly better than FTSC due to its responsiveness to real-time traffic via vehicle detectors, while FTSC remains static with fixed cycle times.

4.2. Effectiveness of the four methods to control signal light

To clarify the effectiveness of the four methods, their performance was compared across key traffic metrics, including average travel time, speed, lost time, and waiting time.

In Figure 8, the DDQN method achieves the lowest mean travel time (188.5s), with a narrow distribution and fewer extreme outliers. This demonstrates its superior ability to optimize signal timings, reduce vehicle delay, and adapt effectively to

dynamic traffic patterns. The DQN method achieves a higher mean travel time (223.7s) than DDQN, though still significantly better than Q-learning, ATSC, and FTSC. Its distribution is relatively compact but includes some high outliers, reflecting occasional inefficiencies caused by overestimation bias in Q-value updates.

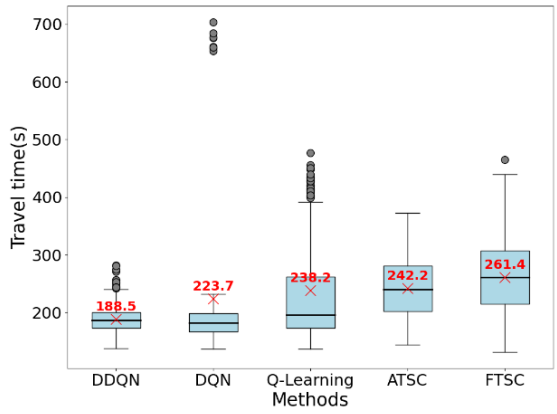

Figure 8. Evaluation of Average Travel Time using different algorithms.

Q-learning performs slightly worse, with an average of 238.2 s and a wider spread of results. The variability reflects the limitations of tabular learning and state discretization, which reduce precision in capturing complex traffic states. ATSC, with a mean travel time of 242.2 seconds, also performed reasonably well by dynamically adjusting signals in response to real-time conditions. However, its reliance on reactive adaptation, rather than long-term optimization, constrains its ability to achieve the same performance improvements as RL approaches. FTSC is a traditional rule-based method that operates on pre-defined timing plans without adapting to real-time traffic dynamics. As a result, it exhibited the weakest performance with the highest mean travel time (261.4 seconds) and a wide distribution.

Table 4 below presents the other metrics of the four traffic signal control methods.

The DDQN method achieves the best overall performance, with the highest average speed (12.14 m/s), the lowest lost time (29.16 s), and the shortest waiting time (18.12 s). This demonstrates the strong capability of DDQN to optimize traffic signal policies by effectively balancing exploration and exploitation while avoiding overestimation bias. The DQN method performs second best, with a slightly lower average speed (11.5 m/s) and significantly higher lost time (63.31 s) and waiting time (43.02 s) compared to DDQN. While still superior to traditional approaches, its performance degradation highlights the limitations of single Q-network estimation. Q-learning performed moderately, with a speed of 10.35 m/s, a lost time of 80.24 s, and a waiting time of 58.91 s. ATSC showed lower speed (9.53 m/s) and higher lost time (77.68 s), though its waiting time (48.57 s) was better than Q-learning. FTSC had the weakest performance, with the lowest speed (8.81 m/s), the highest lost time (101.11 s), and the longest waiting time (69.71 s). Overall, DDQN clearly outperformed the other methods, while FTSC performed the worst.

Table 4. Performance metrics of different traffic signal control methods.

Methods	Speed (m/s)	Lost time (s)	Waiting time (s)
DDQN	11.9	39.16	28.12
DQN	11.5	63.31	43.02
Q-learning	10.35	80.24	58.91
ATSC	9.53	77.68	48.57
FTSC	8.81	101.11	69.71

4.3 Evaluate the influence of hyperparameters in the DDQN model

In this section, we conducted experiments to evaluate the influence of individual hyperparameters in the DDQN model. Firstly, the number of hidden layers and neurons was examined using different values. The experimental results in Figures 9 and 10 indicate that varying the number of hidden layers and neurons in the DDQN architecture does not lead to significant differences in performance. This can be explained as the traffic signal control has a relatively low-dimensional state space (96 states) and a small discrete action space (4 actions). The underlying Q-function is not highly complex, and a shallow network with a moderate number of neurons is already sufficient to approximate it effectively.

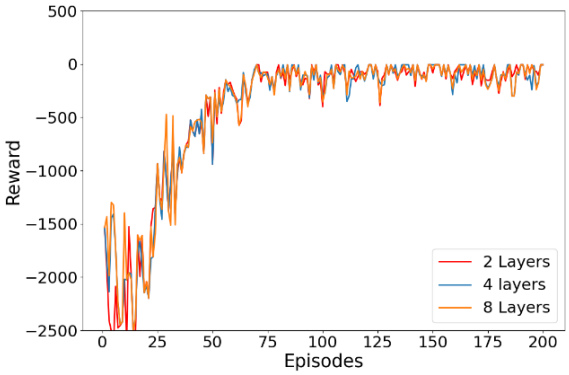

Figure 9. Performance evaluation of DDQN with different hidden layers.

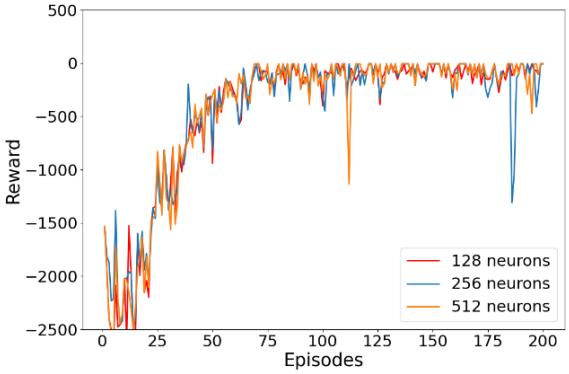

Figure 10. Performance evaluation of DDQN with different neurons in hidden layers.

The learning rate (α) was also tested at different values to examine its effect on convergence speed and stability.

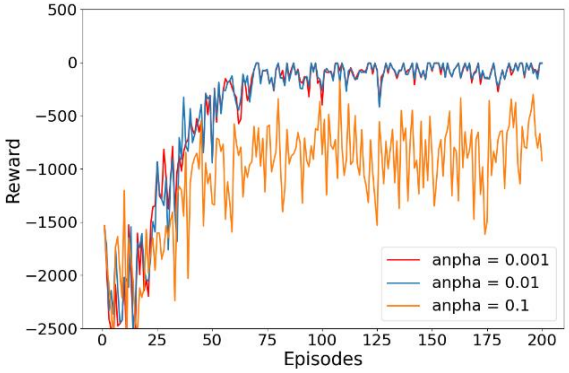

Figure 11. Performance evaluation of DDQN with different learning rates.

Results in Figure 11 showed that higher learning rates accelerated training but often caused unstable policies, while lower values improved stability but slowed convergence. When the learning rate is low, the agent shows stable learning progress, with rewards gradually improving and converging to a relatively high level after around 75–100 episodes. In contrast, a learning rate of 0.1 results in highly unstable training behavior, with rewards oscillating dramatically throughout the entire training process and failing to converge to a satisfactory policy.

5. Discussion and Conclusion

In this study, we propose the use of the DDQN algorithm to manage traffic lights at an isolated intersection, offering an alternative to traditional methods. Through the learning process, our model enhances vehicle performance compared to DQN, Q learning, ATSC, and FTSC scenarios. The results demonstrate the superiority of employing DDQN for traffic signal management at an isolated intersection under heterogeneous traffic. These results align with findings from previous studies.

Our model assumes ideal conditions for the traffic simulation, where vehicle data is collected accurately and promptly. Traffic light control relies on the real-time location data of vehicles within the system, with no interruptions or delays in data transmission. However, our model does not account for other forms of transport, such as buses, bicycles, or pedestrians.

While the simulation results for individual intersections are promising, challenges remain when extending the model to networks of intersections. In such cases, the model becomes more complex, as multiple agents interacting within the same environment can influence each other.

In the future, we aim to apply RL to multi-intersection networks and include a wider variety of traffic participants in our simulations. Additionally, enhancing model performance to achieve faster training times and improved outcomes will be a key focus in future work.

Contributions of authors in this article

Hung Tuan Trinh: Methodology, Data management, Formal analysis, Investigation, Validation, Visualization, Grant acquisition, Feedback on peer review, Writing – original manuscript. **Van Bac-Nguyen**: Data compilation, Data analysis, Investigation. **Duy Quang Tran**: Research Methodology, simulation

Declaration of competing interests and dedication to copyright

The authors declare the absence of any potential conflicts of interest from this study and affirm that the paper has not been previously published.

Data available

Data will not be provided upon request.

References

[1] Anh Khue, "HCMC accelerates transition to green vehicles and reduces pollution," *VnEconomy*, Mar. 8, 2025. [Online]. Available: https://en.vneconomy.vn/hcmc-accelerates-transition-to-green-vehicles-and-reduces-pollution.htm. Accessed: Dec. 2, 2025.

[2] VOV Giao Thông, "Báo động chất lượng không khí ô nhiễm tại TP.HCM," *VOV Giao Thông*, Mar. 8, 2025. [Online]. Available: https://vovgiaothong.vn/bao-dong-chat-luong-khong-khi-o-nhiem-tai-tphcm-d14917.html. Accessed: Dec. 2, 2025.

[3] P. Koonce and L. Rodegerdts, *Traffic Signal Timing Manual*. Washington, DC, USA: Federal Highway Administration, 2008. [Online]. Available: https://nacto.org/wp-content/uploads/2-7_FHWA-Traffic-Signal-Timing-Manual-ch-6_2008.pdf. Accessed: Dec. 2, 2025.

[4] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections," in *Advances in Dynamic Network Modeling in Complex Transportation Systems*, vol. 2, *Complex Networks and Dynamic Systems*. New York, NY, USA: Springer, 2013, pp. 27–66, doi: 10.1007/978-1-4614-6243-9_2.

[5] P. Lowrie, "SCATS: Sydney Co-Ordinated Adaptive Traffic System: A Traffic Responsive Method of Controlling Urban Traffic," Roads and Traffic Authority NSW: Darlinghurst, NSW, Australia, 1990. Available: https://trid.trb.org/View/488852. Accessed: Dec. 2, 2025.

[6] P.B. Hunt, D.I. Robertson, R.D. Bretherton, R.I. Winton , "SCOOT-a Traffic Responsive Method of Coordinating Signals," Transport and Road Research Laboratory (TRRL): Berkshire, UK, 1981. Available: https://trid.trb.org/View/179439. Accessed: Dec. 2, 2025.

[7] P. Kopelias, E. Demiridi, K. Vogiatzis, A. Skabardonis, V. Zafiropoulou, " Connected & autonomous vehicles—Environmental impacts—A review," in *Sci. Total Environ,* vol. 712, 2020. doi: 10.1016/j.scitotenv.2019.135237

[8] A. Olia, S. Razavi, B. Abdulhai, H. Abdelgawad, "Traffic capacity implications of automated vehicles mixed with regular vehicles," *Intell. Transp. Syst,* vol. 22, no. 244-262, 2018. doi: 10.1080/15472450.2017.1404680

[9] C.J. C. H. Watkins and P. Dayan, "Q-learning," Springer Nature Link, Machine Learning, 1992. Volume 8, pages 279–292. doi: 10.1007/BF00992698

[10] V. Mnih, et al, "Playing Atari with Deep Reinforcement Learning," arXiv, Computer Science, Machine Learning, 2013. doi:10.48550/arXiv.1312.5602

[11] H.V. Hasselt, A. Guez, D. Silver, "Deep Reinforcement Learning with Double Q-learning," *arXiv, Computer Science > Machine Learning,* 2015. doi:10.48550/arXiv.1509.06461

[12] R.S. Sutton,A.G. Barto, "Reinforcement Learning: An Introduction", London, England: The MIT Press Cambridge, Massachusetts, 2015. Available: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf. Accessed: Dec. 2, 2025.

[13] R.E. Bellman and S.E. Dreyfus, "Applied dynamic programming", Santa Monica: RAND Corporation, 1962. Available: https://www.rand.org/content/dam/rand/pubs/reports/2006/R352.pdf. Accessed: Dec. 2, 2025.

[14] B. Michael et al, "SUMO – Simulation of Urban MObility: An Overview," in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*, 2011. Available: https://elib.dlr.de/71460/. Accessed: Dec. 2, 2025.

[15] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142 ,* 2016. doi: 10.48550/arXiv.1611.01142

[16] X. Glorot, A. Bordes, Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. Available: https://proceedings.mlr.press/v15/glorot11a.html. Accessed: Dec. 2, 2025.

[17] D.P Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization," in *International*

Conference on Learning Representations (ICLR), 2015. https://doi.org/10.48550/arXiv.1412.6980

[18] S. Krauss, Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics, Cologne, Germany: Ph.D. Thesis, University of Cologne, , 1997. Available: https://www.osti.gov/etdeweb/biblio/627062. Accessed: Dec. 2, 2025

[19] J. Erdmann, "Lane-Changing Model in SUMO," in *the 2nd SUMO User Conference (SUMO2014)*, Germany, 2014. Available: https://elib.dlr.de/102254/1/Springer-SUMOs_Lane_changing_model.pdf. Accessed: Dec. 2, 2025

[20] H. Kaths, A. Roosta, "A Framework for Simulating Cyclists in SUMO," in *SUMO User Conference 2023*, Wuppertal, Germany, 2023. https://doi.org/10.52825/scp.v4i.219