

Article

# Optimization Control of Adaptive Traffic Signal with Deep Reinforcement Learning

Kerang Cao <sup>1</sup>, Liwei Wang <sup>1</sup>, Shuo Zhang <sup>1</sup>, Lini Duan <sup>2</sup>, Guimin Jiang <sup>3</sup>, Stefano Sfarra <sup>4</sup>, Hai Zhang <sup>5</sup> and Hoekyung Jung <sup>6,\*</sup>

- <sup>1</sup> Key Laboratory of Intelligent Technology of Chemical Process Industry in Liaoning Province, College of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang 110142, China; caokerang@syuct.edu.cn (K.C.); victorywlw@163.com (L.W.); syzs1210@163.com (S.Z.)
- <sup>2</sup> Big Data Management and Application, Shenyang University of Chemical Technology, Shenyang 110142, China; liniduan@163.com
- <sup>3</sup> School of Automation and Electrical Engineering, Shenyang Ligong University, Shenyang 110159, China; 22bg18171@stu.hit.edu.cn
- <sup>4</sup> Department of Industrial and Information Engineering and Economics, University of L'Aquila, I-67100 L'Aquila, Italy; stefano.sfarra@univaq.it
- <sup>5</sup> Centre for Composite Materials and Structures (CCMS), Harbin Institute of Technology, Harbin 150001, China; hai.zhang.1@ulaval.ca
- <sup>6</sup> Computer Engineering Department, Paichai University, Daejeon 35345, Republic of Korea
- \* Correspondence: hkjung@pcu.ac.kr

**Abstract:** The optimization and control of traffic signals is very important for logistics transportation. It not only improves the operational efficiency and safety of road traffic, but also conforms to the direction of the intelligent, green, and sustainable development of modern cities. In order to improve the optimization effect of traffic signal control, this paper proposes a traffic signal optimization method based on deep reinforcement learning and Simulation of Urban Mobility (SUMO) software for urban traffic scenarios. The intersection training scenario was established using SUMO micro traffic simulation software, and the maximum vehicle queue length and vehicle queue time were selected as performance evaluation indicators. In order to be more relevant to the real environment, the experiment uses Weibull distribution to simulate vehicle generation. Since deep reinforcement learning takes into account both perceptual and decision-making capabilities, this study proposes a traffic signal optimization control model based on the deep reinforcement learning Deep Q Network (DQN) algorithm by considering the realism and complexity of traffic intersections, and first uses the DQN algorithm to train the model in a training scenario. After that, the G-DQN (Grouping-DQN) algorithm is proposed to address the problems that the definition of states in existing studies cannot accurately represent the traffic states and the slow convergence of neural networks. Finally, the performance of the G-DQN algorithm model was compared with the original DQN algorithm model and Advantage Actor-Critic (A2C) algorithm model. The experimental results show that the improved algorithm increased the main indicators in all aspects.



**Citation:** Cao, K.; Wang, L.; Zhang, S.; Duan, L.; Jiang, G.; Sfarra, S.; Zhang, H.; Jung, H. Optimization Control of Adaptive Traffic Signal with Deep Reinforcement Learning. *Electronics* **2024**, *13*, 198. <https://doi.org/10.3390/electronics13010198>

Academic Editor: Mohamed Karray

Received: 27 October 2023

Revised: 6 December 2023

Accepted: 6 December 2023

Published: 2 January 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Traffic signal optimization control is a method of the intelligent management of traffic facilities such as intersections and road sections through technological means, in order to improve the operational efficiency, safety, and comfort of road traffic. It can improve logistics efficiency, reduce costs, and improve delivery reliability, while also being in line with the trend of green environmental protection and information technology development; in addition, it has a positive role in promoting the sustainable development of the logistics

industry [1]. The proposal of the concept of intelligent transportation can effectively enhance the capacity of road transportation. Smart transportation combines technologies such as the Internet of Things, cloud computing, big data, and artificial intelligence to gather traffic information through high technology and support the control of traffic management, transportation, and other transportation fields in order to fully guarantee traffic safety, bring into play the effectiveness of transportation infrastructure, and improve the management of transportation systems for smooth public travel and economic development [2]. Among them, intelligent optimization of traffic signal control is an important application. Most traditional traffic signal optimization control systems use traditional fixed timing schemes, which cannot be adjusted according to the real-time status of intersections and are prone to traffic congestion if the traffic flow is too high. Therefore, many scholars have started to study Adaptive Traffic Signal Control (ATSC), such as developing new adaptive signal optimization control systems that set different signals according to different geographical areas [3]; using pressurized routing algorithms in the field of communication to effectively reduce traffic congestion and decrease the average travel time of vehicles on the road [4]; introducing greedy ideas and modeling two-dimensional automata matrices for vehicles so that they can explore rerouting paths in congested areas [5]; and setting up new traffic light schedules that take holiday factors into account to model the behavior of traffic lights [6]. However, all these ATSC systems rely on manually designed traffic signal schemes, which have many limitations.

With the development of artificial intelligence and IoT technologies, intelligent traffic light signal optimization control methods are becoming more and more sophisticated. Compared with traditional traffic signal optimization control methods, intelligent traffic signal optimization control methods are able to adjust the strategy according to real-time road conditions. Since the traffic signal optimization control problem is a sequential decision problem, reinforcement learning is well suited to solve the problem. Noaeen et al. [7] summarized the applications of reinforcement learning in various areas of traffic signal optimization control over the last 26 years, exploring all the applied methods and defining the main first events in the scope, and finally giving suggestions for development. KuangLi et al. [8] simplified the representation of the states using a generated Q-table, and proposed a bi-objective reward function to enable an intelligent body to quickly learn the optimal signal timing strategy. Müller et al. [9] argued that the deployment of reinforcement learning for traffic signal optimization control lacks safety in realistic intersections, so they designed a safe action space and action masking mechanism to ensure safety in practical applications, driving the application of reinforcement learning in realistic scenarios. Subsequently, Zhao et al. [10] argued that existing reinforcement learning-based traffic signal optimization control progressed slowly due to poorly designed states and rewards in reinforcement learning, so they introduced the concept of strength, which ensured that their states and rewards reflected vehicle states more accurately. Genders et al. [11] compared two reinforcement learning adaptive traffic signal controllers with the Webster controller to demonstrate the superiority of reinforcement learning, by developing a new action selection strategy. Zhang et al. [12] combine reinforcement learning with vehicle wireless technology. Vehicle wireless technology uses vehicle-to-infrastructure (V2I) wireless communication to detect vehicles, but traditional V2I intelligence detects vehicles equipped with wireless communication capabilities, and reinforcement learning gives help to this problem. Boukerche et al. [13], to address the problem that existing methods ignore the impact of transmission delay on the system exchanging traffic flow information, proposed a traffic state detection method, and proved to solve the data transmission delay problem by an experimental comparison. Alegre et al. [14] proposed the TOS( $\lambda$ )-FB algorithm and proved its efficiency by combining the Fourier basis function and the reinforcement learning SARSA( $\lambda$ ) algorithm in order to solve the dimensional explosion problem due to the large state space. Wang et al. [15] investigated multi-intelligent reinforcement learning for large-scale traffic signal optimization control problems. A multi-intelligent reinforcement learning algorithm called Co-DQL was proposed with improved local state sharing

methods and reward allocation mechanisms to improve the robustness of the learning process. Antes et al. [16] optimized the problem of passing information between intelligences in a multi-intersection traffic environment. They used a hierarchical approach to increase the basis for action selection of the intelligences and proposed an information-up, recommendation-down framework to organize reinforcement learning. Experiments show that this approach outperforms the fixed timing approach and the reinforcement learning approach without layering. However, when the amount of data increases or the state space is high-dimensional, the computational and learning capabilities of the above-mentioned reinforcement learning methods without using neural networks are severely limited.

With the rapid development of deep learning, the application of deep reinforcement learning is becoming more and more widespread [17], which can be a good solution to the shortcomings of reinforcement learning. Park et al. [18] developed a traffic signal optimization control model for a single intersection and two coordinated intersections, and used DQN and Synchro 6.0 software optimized fixed-time timing for comparison to demonstrate the effectiveness of deep reinforcement learning algorithms. Zhao et al. [19] used a combination of the convolutional neural network (CNN) and Recurrent Neural Network (RNN) with an additional layer of Long and Short-Term Memory neural network (LSTM) in the outer layer of CNN by proposing a new Deep Recurrent Q-Learning (DRQN) algorithm able to define the reinforcement learning states as matrices to better characterize the environment. Wan et al. [20], on the other hand, used a CNN model to compare with a DNN model experiments and modeled traffic intersections using VISSIM simulation software. The experiments proved that the CNN model has a better ability to extract features than the DNN model. Zou et al. [21] combined Bayesian methods with reinforcement learning to solve the overfitting problem of neural networks and to speed up the training of neural network models. Chu et al. [22] used for the first time a three-dimensional reinforcement learning state; the data were taken using photos by intersection cameras and vehicles were visualized using Sumo-web3d 3D visualization tool. To better extract data features, experiments were conducted using three algorithms and two combinations of neural network model permutations, and the results showed that DQN + CNN worked best. Muresan et al. [23] used the idea of migration learning to apply the model training method for a single intersection to multiple intersections and simulated the real scenario using SUMO simulation software. Alhassan et al. [24] used the DQN algorithm to perform traffic optimization on four different SUMO-simulated streets and re-planned the streets. Bouktif et al. [25] combined a dual deep Q-network (DDQN) with a priority experience replay (PER) mechanism and proposed a new method for defining states and rewards. They adopted this method to optimize the agent's strategy for selecting actions, allowing the agent to select the best action faster. Su et al. [26] believed that the scalability of existing optimization algorithms for signal light control was bad, and the current deep reinforcement learning algorithms had the problem of overestimating the Q-value. Therefore, they reduced the dimensionality of the action space and proposed a dynamic delayed update method based on Exponential Weighted Moving Average (EWMA) to optimize the algorithm and improve the performance of the model. Due to the increasingly widespread application of neural networks, they are being used in more than just transportation [27,28]. This proves the widespread application of neural networks in various industries.

In response to the above findings, it was found that the original DQN algorithm using DNN to represent the Q function was not as effective as CNN and RNN, and CNN was the most effective. Many studies overlook the detailed description of the traffic environment, which can affect the convergence speed of neural networks. This also leads to the inability of ordinary structured CNN to adapt well to complex traffic situations. This article starts with the design of reinforcement learning states, then improvement of the CNN structure, and adjustment of action selection strategies. Thus, the main contributions of this paper, based on the previous academic studies, are the following:

- (1) By improving the action selection strategy of deep reinforcement learning, the model can be trained faster, and the number of vehicle queues can be effectively reduced by experimental verification.
- (2) Based on the DQN algorithm, the G-DQN algorithm is proposed. The algorithm redefines the state and reward in the three elements of reinforcement learning, and improves the original CNN network structure into a dual-channel CNN, so that it can reflect the state of the traffic intersection more accurately and improve the performance of the model, and its optimization effect is proved to be better than the DQN algorithm, the A2C algorithm and the fixed signal timing strategy through comparison experiments.
- (3) The experiments are modeled and simulated by SUMO simulation software, and the Weibull distribution is used to simulate the traffic flow in the real environment.

The composition of the remaining part of the paper can be summarized as follows. Section 2 provides a detailed introduction of the basic tools and methods used in the study. Section 3 explains the construction and implementation of the G-DQN model. Section 4 explains the method of data generation and the way of conducting the experiments to demonstrate the universality and superiority of the proposed G-DQN in two optimization objectives by comparing it with the original DQN algorithm, A2C algorithm, and current fixed signal timing methods. Finally, in Section 5, the research conclusions and innovative directions for future works are presented.

## 2. Related Work

This section introduces SUMO 1.14.1 simulation software and deep reinforcement learning methods, providing theoretical support for the following experiments.

### 2.1. SUMO Traffic Simulation Software

SUMO (Simulation of Urban Mobility) is an open source, microscopic, multi-modal traffic simulation software. Its microcosm is reflected in the fact that each vehicle is explicitly modeled with its own individual route of movement in the road network structure. It allows to simulate how a complex traffic flow consisting of many individual vehicles moves through a given road network structure after a given traffic demand. The simulation software solves a large number of traffic management problems by simulating vehicle movements for different vehicle types at spatially continuous and temporally discrete times, and by managing road network structures of up to 20 lanes in both directions, supporting lane changes, multi-lane streets, and different access rules. SUMO has a fast OpenGL graphical user interface that enables the user to design each element of each roadway at an intersection, including the number of roadway strips, directions, and the number of lanes, the location of traffic signals and the phase sequence and duration of traffic signals. In addition, the package Traci in SUMO can easily enable it to communicate with python, which facilitates the simulation. By calling Traci, various data from the traffic simulation, such as vehicle waiting time, number of vehicles on the road, and vehicle speed, can be obtained in real time. From this, the real-time traffic status of the intersection can be obtained and the optimization control strategy of the signal can be adjusted [29].

In SUMO, a simulation requires several configurations: a road network file (.net.xml), a traffic flow file (.rou.xml), an additional file (.add.xml), and an executable file (.sumocfg). Among them, road network files, traffic flow files, and executable files are indispensable. The working principle of SUMO simulation software is as follows:

- (1) Create a road network file. A road network file is used to generate the scenarios required for the experiment, including various parameters of the intersection, such as the number of lanes, lane distribution, lane length and transition rules. It has three ways to generate these. The first is to write the code manually. The second is to design through SUMO's own netedit road network editor. The Netedit road network editor can be very convenient for users to directly draw any shape of road network in the editor. The signal light control phase can also be generated by the editor. The third is

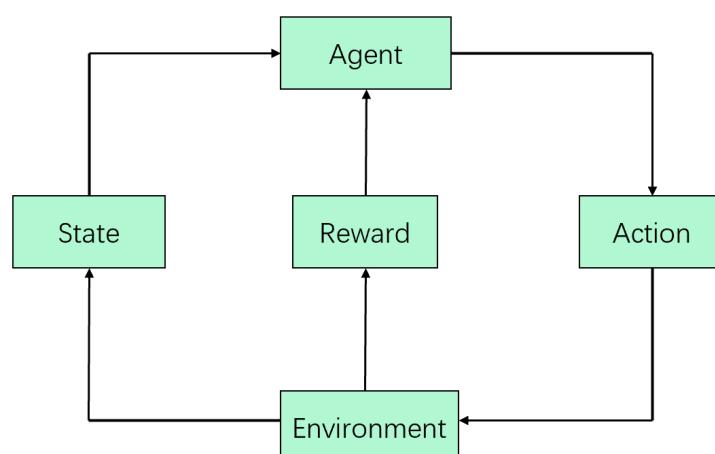
to download the real-world road network through OpenStreetMap to generate the osm file, and then convert it into a road network file.

- (2) Create a traffic flow file. A traffic flow file is used to generate vehicles and their attributes, including various parameters of vehicles, such as vehicle ID, vehicle type, vehicle color, and lanes entered and left. The traffic file is intelligently generated by manually writing code.
- (3) Create an executable file. An executable file should be configured with the file names of the two files mentioned above, so that the road network and traffic flow can be combined for simulation.
- (4) Start simulation. Open sumo-gui in the bin directory and open the configured executable file to start the simulation.

## 2.2. Deep Reinforcement Learning

Deep reinforcement learning (DRL) has become one of the hot spots in artificial intelligence research. DRL comes from deep learning that has been applied in the recent past in several research fields [30–34]. DRL has also been widely used in various fields such as recommender systems, robot control, end-to-end control, and natural language dialogue systems. It integrates the perceptual ability of deep learning and the decision-making ability of reinforcement learning to first explore and optimize the system objectives through reinforcement learning, and then, second, to solve the optimization control strategy problem through deep learning algorithms. This is with the aim to understand the dimensional explosion problem caused by the lack of storage space of reinforcement learning algorithms, so as to optimize the objective function of the system and seek the optimal value of the system.

As shown in Figure 1, in the process of reinforcement learning, the agent and the environment are constantly interacting, which is a sequential decision problem. Generally, reinforcement learning problems are regarded as Markov decision processes (MDPs). MDPs are a mathematical model for sequential decision making, which include a set of interactive objects such as environment and agent. In addition, they also includes five model elements: state, action, reward, strategy, and value function. Based on these elements, MDPs calculate the optimal strategy, which is how to select the best action in each state to maximize the expected cumulative reward [35]. At each moment the intelligence receives a state from the environment, and based on such a state, the intelligence makes a corresponding action, which then acts on the environment and gives the intelligence a reward and the next state, and the intelligence selects the action for the next state based on the effect of the reward on the strategy. By continuously cycling the above process, the optimal strategy to achieve the goal is finally obtained. The value function is divided into a state value function and an action value function, which evaluate the state and action, respectively, and their results are the expectation of cumulative rewards.

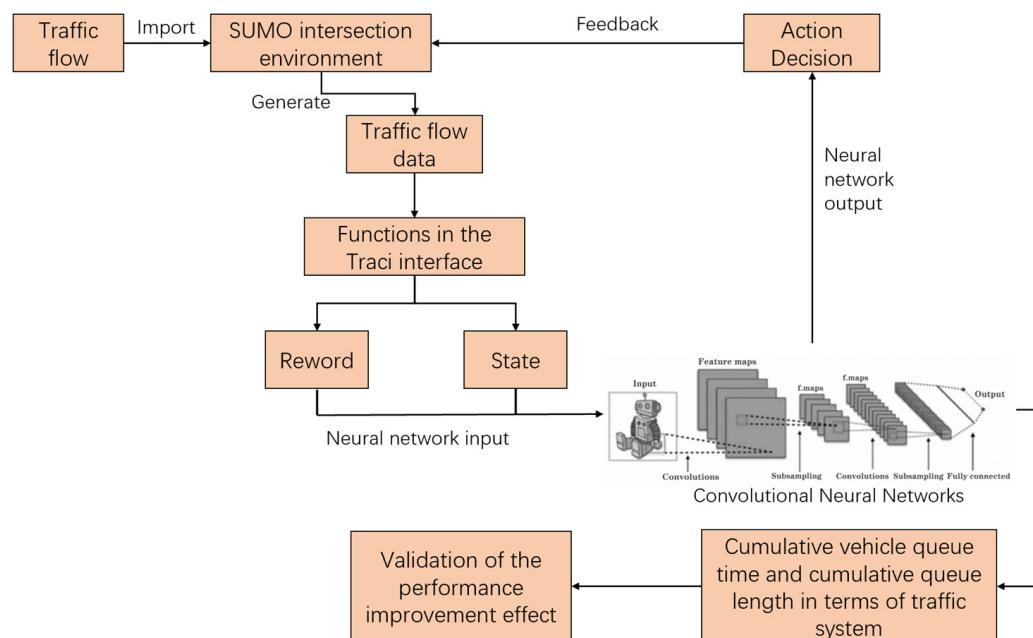


**Figure 1.** Intensive learning process.

The neural network in deep learning has strong feature extraction capability, which can process features more easily and no longer requires manual feature extraction. Especially in large-capacity data processing, the effect of deep learning algorithms has obvious advantages over traditional artificial intelligence algorithms.

This study applies a convolutional neural network, as shown in Figure 2. The learning process of deep reinforcement learning can be described as follows:

- (a) The established single-intersection training scenario is implanted into SUMO simulation software to form SUMO's intersection environment and we use Weibull distribution to simulate vehicle generation and generate traffic flow data.
- (b) The traffic flow data are processed using the functions under the vehicle module in the Traci interface to calculate the state and reward, the Q-value is calculated from the reward, and the state and Q-value are passed into the convolutional neural network for training.
- (c) The convolutional neural network will use the output generated by each turn prediction, i.e., action, for decision making until the specified training turn is reached, and it will feed back to the SUMO intersection environment to change the phase of the signal in SUMO. In addition, the queue length and queue time of vehicles in the traffic system for each round will be counted.
- (d) When the convolutional neural network completes the prescribed training rounds, the queue length and queue time of vehicles in the traffic system for each round are compared to verify the performance improvement in the improved algorithm.



**Figure 2.** Technology roadmap.

Since the value function-based reinforcement learning algorithm is more suitable for dealing with continuous state space and discrete action space problems, this study uses the DQN algorithm as the basis for improvement, so as to perform intelligent optimization of regional traffic signal optimization control. The naming table of variables is shown in Table 1.

**Table 1.** Nomenclature table.

Parameter Name	Parameter
State	s
Action	a
Reward	r
Time step	t
Action selection strategy	$\pi$
Neural network parameters	$\theta$
Experience pool size	M
Learning rate	$\alpha$
Discount factor	$\gamma$
Target network update period	C

The DQN algorithm is an improved algorithm based on the q-learning algorithm of reinforcement learning, which uses neural networks in deep learning to record the action values in each state. The input of the network is the state information and the output is the value of each action to optimize the action decision in the next state. the DQN algorithm contains a  $Q$ -value function and a  $V$ -value function, and the algorithm uses the  $Q$ -value function to evaluate the goodness of the current state it is in. The  $Q$ -value function is the value of taking an action at with a state shift probability of P in the current state  $s_t$  under a specific strategy  $\pi$ , and the  $V$ -value function is the average value in the current state  $s_t$ . The  $Q$ -value function is calculated as Equation (1) and the  $V$ -value function is calculated as Equation (2) as follows [36]:

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P_{s_t \rightarrow s_{t+1}}^{a_t} [r(s_t, a_t) + \gamma V(s_{t+1})] \quad (1)$$

$$V_\pi(s_t) = \sum_{a_t} \pi(s_t, a_t) \sum_{s_{t+1}} P_{s_t \rightarrow s_{t+1}}^{a_t} [r(s_t, a_t) + \gamma V(s_{t+1})] \quad (2)$$

where  $Q_\pi(s_t, a_t)$  is the  $Q$ -valued function, and  $V_\pi(s_t)$  is the  $V$ -valued function;  $\gamma$  represents the degree to which the agent is concerned about future benefits, and a larger value indicates that the agent is more concerned about future benefits.

The DQN algorithm adds experience replay and objective network mechanisms [31]. The experience replay mechanism can store the experience (consisting of five parameters: current state  $s_t$ , action  $a_t$ , reward  $r$ , next-turn state  $s_{t+1}$ , and whether to reach the scheduled turn True/False) in the experience pool, and then each time the input of the neural network is randomly sampled from the experience pool, which can break the correlation between the data and make the data satisfy independent identical distribution, thus reducing the parameter update. This can break the correlation between the data and make the data meet independent homogeneous distribution, thus reducing the variance in the parameter update and improving the convergence speed of the network. Meanwhile, the target of the network mechanism is to build a network with the same structure on the basis of the original network, but the time to update the parameter  $\theta$  is different.  $\theta$  is the weight parameter of the convolutional neural network model. A continuous function  $Q_\theta(s, a)$  with parameter  $\theta$  is established in the DQN algorithm to approximate  $Q_\pi(s, a)$ , and  $Q(s_t, a_t)$  is calculated from the current state  $s_t$ , action  $a_t$  as the output result of the network  $r + Q(s_{t+1}, a_{t+1})$  is the target value, and the iterative update Formula (3) is as follows:

$$Q(s_{t+1}, a_{t+1}) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s, a)] \quad (3)$$

Since the parameters of the original network are updated in real time during the training process, while the target network parameters and the target value  $r + Q(s_{t+1}, a_{t+1})$  are constant, the introduction of the target network increases the stability of learning. These two mechanisms are a good solution to the problems of neural networks that cannot guarantee convergence, training instability and training difficulties.

In addition, the DQN algorithm also has a loss function to calculate the error between the predicted value of the intelligent agent and the actual value. The definition of loss function L is shown in Formula (4):

$$L = [r + \gamma \max Q(s_{t+1}, a_{t+1}; \theta') - Q(s, a; \theta)]^2 \quad (4)$$

The loss function can help the agent learn more accurate Q values, promote their exploratory behavior, and avoid the overfitting of neural networks.

### 3. Traffic Signal Optimization Control Model

In the reinforcement learning framework, the intelligent body is always interacting with the environment. The intelligent body contains three elements, i.e., state, action and reward. The definition of the elements directly affects the performance of the algorithm, and how to define the elements is the key to build the traffic light optimization control model. The following is the definition of the environment, the three elements of reinforcement learning, the learning strategy and the convolutional neural network in the G-DQN algorithm.

#### 3.1. Environment

The scenario used in this study is shown in Figure 3, which is an intersection with eight lanes in both directions. The traffic flow file is generated using a Weibull distribution, where each generated car is assigned a random origin and destination. The Weibull distribution is a continuous probability distribution function with an overall increase followed by a decrease, which fits well with the pattern of high and low traffic peaks. The environment for the scenario is defined as follows:



**Figure 3.** SUMO interface.

**Lanes:** The lanes of an intersection are divided into an approaching lane and an exiting lane. The lane in which the vehicle is approaching the intersection is the entry lane, and the lane in which the vehicle is exiting the intersection is the exit lane, with four exit lanes and four entry lanes in the same direction. The four entry lanes consist of two straight

lanes, one left-turn lane and one right-turn lane, and the vehicle movements are randomly assigned to straight, left-turn and right-turn movements.

Traffic signals: In the traffic movement, each approach lane to the corresponding exit lane has a corresponding traffic signal to direct the vehicle movement. Among them, the red light represents the prohibition of vehicle movement, the green light represents the permission of vehicle movement, and the yellow light represents the buffer between the red and green lights, and the vehicles that have crossed the stop line can continue to pass. In general, right turns are not subject to traffic signals.

Phase: Phase represents the set of all non-conflicting signal combinations at an intersection, each representing the state of passage at the intersection at that moment. For example, the east–west direction allows straight ahead and right turn, but not left turn; at this time the traffic signal for the east–west straight ahead lane is green, while the signal for the left turn lane is red.

Phase sequence: Phase sequence refers to the combination of several phases in the phase set, which is used to control the sequence of phase changes and is the strategy adopted by the intelligence after learning, and is also the main optimization goal.

The problem of traffic signal optimization control can be defined as the intelligent body as an intersection, the state  $s$  of the intersection, the action  $a$  selected by the intelligent body from a predefined set of actions, the reward  $r$  derived from the reward function, the time step  $t$ , the policy  $\pi$ , and the value function  $Q_\pi(s,a)$  under that policy. First, the intelligence perceives the environment and receives the current state  $s_t$ . Then, the next action  $a_t$  is selected based on the current state  $s_t$  and previous experience, and the selected action yields the reward  $r_t$  and the value function  $Q_\pi(s_t,a_t)$  for this time. The value of this action is then evaluated by the reward  $r_t$  and the value function  $Q_\pi(s_t,a_t)$  for this time. Finally, these experiences are used to train the convolutional neural network, which updates itself by continuously receiving states and rewards from the environment and can dynamically select the phase sequence of traffic signals by learning from historical experiences. However, the traffic signals must follow the corresponding rules of operation, and the duration of the phases in this study model is fixed and can be adjusted according to the experimental effects. Within the same time step, there must be lanes in the phase that are in the green or yellow light state, and there cannot be a state where the entire intersection is red.

The design process of the simulation environment is as follows:

- (1) Construct a bidirectional eight lane intersection environment in netedit of SUMO simulation software.
- (2) Use Python to write classes that generate vehicles. This function uses the Weibull distribution function to generate traffic flow files. It ensures that the vehicles in the environment are randomly generated.
- (3) Write simulation classes through the Traci interface in Python. Traci is an interface between SUMO and Python, and its built-in functions can help Python set or obtain information in SUMO. For example, `traci.start()` can call SUMO to start simulation; `traci.vehicle.getIDList()` can be used to obtain the vehicle ID; `traci.vehicle.getRoadID()` can be used to obtain the road ID; `traci.vehicle.getSpeed()` can be used to obtain the vehicle speed; `traci.vehicle.getAccumulatedWaitingTime()` can be used to obtain the cumulative waiting time of the vehicle; `traci.trafficlight.setPhase()` sets the phase of the signal light; `traci.vehicle getPosition()` can be used to obtain vehicle position; etc. We use these functions to control and obtain information from simulation software.
- (4) Import the simulation environment code into the deep reinforcement learning algorithm and connect it with the neural network model framework for experiments.

### 3.2. State

The state in deep reinforcement learning is used to describe the current characteristics of the environment, which is generally used as the input of the neural network to provide the basis for the decision of the intelligence. In related studies, the state of an intersection is

usually represented by a matrix as input to a convolutional neural network in order to better represent the features of the intersection and to minimize the training time of the neural network. First is the vehicle location matrix: since the congestion level of the intersection depends mainly on the approach lanes, the four approach lanes in each direction of the intersection are divided into 10 cells of the same size; i.e., a  $16 \times 10$  matrix can be generated. If a vehicle exists in the cell, the state is set to 1; otherwise, it is 0. Secondly, the speed matrix: the speed matrix is also a  $16 \times 10$  matrix, and the data in the speed matrix are the average speed of all vehicles in the cell. In addition, to facilitate convolutional neural network training, the speed matrix is normalized and all the data in the matrix are divided by the road speed limit.

It is indeed a simple and efficient way to model the intersection by discretizing the approaching lanes into cells, but for the actual road conditions, the closer the vehicles are to the intersection, the greater the impact on the current traffic environment. Therefore, in order to more accurately reflect the state of the intersection, this paper introduces the idea of cluster series. The cluster series originates from the Warring States “Sun Bin Art of War—Ten Array”, which is a series of numbers  $\{a_n\}$  grouped according to a certain regular geometric pattern. The common arrangements of numbers are from small to large order, an “S” arrangement, the Yang Hui triangle arrangement, and other forms. The common grouping forms are triangle, rectangle, trapezoid, sawtooth, a long snake formation form, etc. For example, given a series  $\{a_n\}$ :  $a_1, a_2, a_3, a_4 \dots$  in accordance with the rules of the equal difference series tolerance of one grouping order, then you can obtain the sequence of the group as a unit:  $(a_1), (a_2, a_3), (a_4, a_5, a_6) \dots$  In order to reflect the impact of the vehicle distance from the intersection on the road conditions, it will drive into the lane. The 10 cells  $\{a_{10}\}$  delineated (in order from near to far from the intersection) are divided into four groups, namely  $(a_1), (a_2, a_3), (a_4, a_5, a_6), (a_7, a_8, a_9, a_{10})$ ; for the reasonableness of weight assignment and convenience of neural network training, the normalized matrix should be kept as normalized as possible, so the values in it are assigned weights 0.4, 0.3, 0.2, and 0.1, respectively, indicating the degree of influence on road conditions.

### 3.3. Action

Action in deep reinforcement learning is the interaction between the intelligence and the environment, determined by the current state and the reward of the feedback. In order to allow the traffic to pass through the intersection as soon as possible and to improve the efficiency of vehicular traffic at the intersection, the actions defined in this paper caused the signal of a set of lanes to turn green in one cycle. All possible actions are defined as a set of phases:

$$A = \{EW, EWL, SN, SNL\} \quad (5)$$

where  $EW$  is the east–west-direction-driving vehicles that can go straight and turn right;  $EWL$  is the east–west-direction-driving vehicles that can only turn left;  $SN$  is the north–south-direction-driving vehicles that can go straight and turn right; and  $SNL$  is the north–south-direction-driving vehicles that can only turn left. Since the phase sequence is the main optimization target, in this study we set the duration of green and yellow lights as fixed values, which were set to 12 s and 4 s, respectively, and the remaining time was the duration of red lights. When the intelligent body selected the action, if the two actions before and after were consistent, it would continue to maintain the original signal light; if not, it would add a yellow light to buffer between the switching of red and green lights.

### 3.4. Reward

The goal of reinforcement learning intelligence is to maximize the total reward that can be obtained when the learning is completed, so the setting of the reward function is particularly important, which is the learning direction of reinforcement learning. In previous studies, most models simply designed the reward function as the change in cumulative waiting time between adjacent cycles [37]. This approach has appeared in many previous studies, and using only a single cumulative waiting time change cannot accurately

describe the traffic environment in the environment. The variables in the reward function need to be closely related to the optimization objective of the model, so the reward function was defined in this paper as a weighted linear combination of the change in cumulative vehicle waiting time  $T$  and the cumulative duration of the yellow light  $Y$  as the intersection road conditions deteriorate. Both indicators reflect the effect of the model to improve traffic after learning. The value of  $T$  is the waiting time of all vehicles in the previous time step minus the waiting time of all vehicles in the current time step, which is negative when the road conditions become worse. The  $Y$  value is the duration when the signal light switches to yellow. Due to the high traffic flow at the intersection studied in this article, the longer duration of the traffic lights can improve the passing rate of vehicles. Frequent phase switching can easily lead to vehicle congestion. Finally, the reward function can be expressed as follows:

$$R = k_1 T + k_2 Y \quad k_1 + k_2 = 1, k_2 < k_1 \quad (6)$$

where  $k_1$  and  $k_2$  are the weights of  $T$  and  $Y$ , respectively, and the size of  $k_1$  and  $k_2$  is constrained because of the large influence of  $T$  on the optimization effect.

### 3.5. Action Selection Strategy

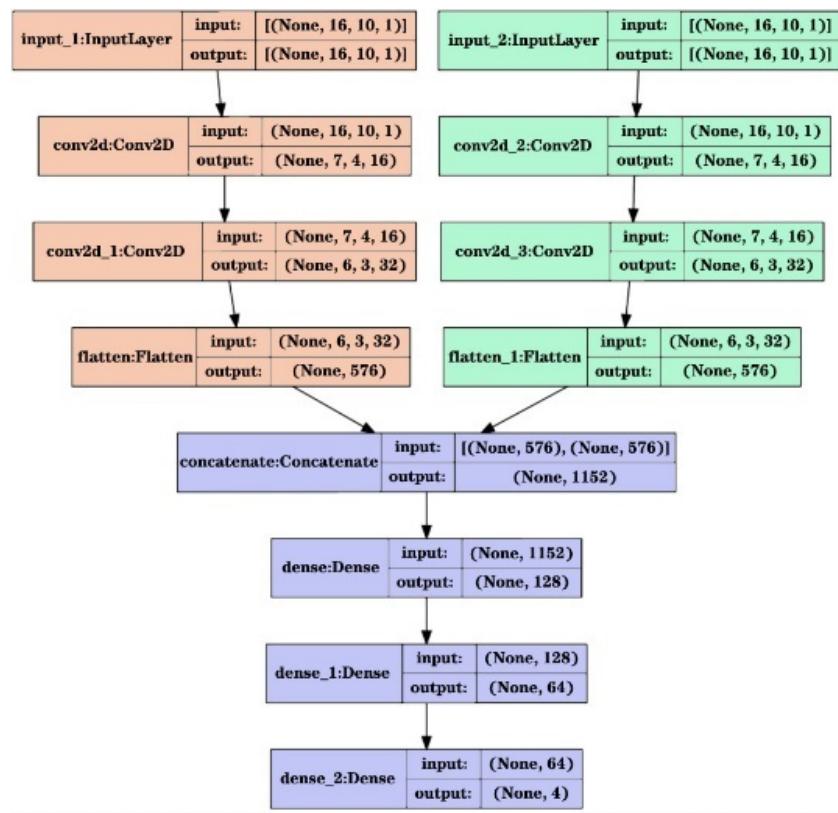
In order to prevent the neural network training from always training in the direction of the maximum value function, i.e., the intelligent body always uses the action that maximizes the reward function when making action decisions, which leads to overfitting the neural network, in this paper we used  $\varepsilon$ -greedy as the action selection strategy. The idea is that the intelligence selects the action with the greatest known reward with the probability of  $1 - \varepsilon$  at each decision, called exploitation, and randomly selects an action from the set of actions with probability of  $\varepsilon$ , called exploration:

$$a = \begin{cases} \underset{a}{\operatorname{argmax}} Q(s, a), 1 - \varepsilon \\ \text{random}(A), \varepsilon \end{cases} \quad (7)$$

The purpose of this strategy is to prevent the neural network from overfitting caused by training in the direction of maximum  $Q$  value all the time after it starts training, which can enable the intelligence to choose the appropriate learning strategy in a limited number of rounds and achieve a balance between exploitation and exploration to obtain the maximum cumulative reward. Although the strategy does play a certain optimization role for the neural network, the direction of action selection is basically fixed after the neural network is trained for many rounds, and too high a  $\varepsilon$  value will affect the convergence speed of the model. Therefore, in this paper,  $\varepsilon$  was dynamically adjusted during the training of the neural network, and was set to 1 at the beginning of the training, and then decreased according to the number of training rounds. This avoided the overfitting of the neural network and also sped up the convergence of the model.

### 3.6. Convolutional Neural Network

Since the deep reinforcement learning state of this study extracted two feature matrices, a dual-channel CNN design was adopted in order for the neural network to obtain more information about the traffic intersection. The dual-channel CNN obtained double the local features by different inputs, which could describe the current traffic state more accurately compared with the single-channel CNN. The model structure is shown in Figure 4. Firstly, the model contained two parallel networks with input position and speed matrices, respectively, which underwent feature extraction by two convolutional layers; then, the Flatten layer flattened the refined features; the concatenate layer was responsible for uniting the features; and finally the fully connected layer was added to integrate and extract the data.



**Figure 4.** Convolutional neural network structure.

The network uses relu activation function for all convolutional and fully connected layers except the last fully connected layer, which uses liner activation function to output a traffic light signal that matches the action space.

The G-DQN algorithm process is shown in Algorithm 1:

---

**Algorithm 1** Steps of G-DQN algorithm.

---

Input

Initialize the number of training rounds episodes, the number of training steps per round  
Initialize the optimization network Q and the parameter  $\theta$ . The target network  $Q' = Q$  and the parameter  $\theta'$

Initialization batch, experience pool size M, learning rate  $\alpha$ , discount factor  $\gamma$ , learning strategy probability  $\epsilon$ , target network update period C

Start

For each episode:

Initialize the environment and divide the lanes into groups to obtain the state s

For each step of episode:

Select an action a randomly with probability  $\epsilon$  or with probability  $1 - \epsilon$   $a = \text{argmax}_a Q(s, a)$ ,  
 $\epsilon = 1 - (\text{episode}/\text{episodes})$

Execute action a to get new state  $s'$  and reward r

Store  $(s, a, r, s')$  in the experience pool

Get the batch group  $(s, a, r, s')$  from the experience pool

target  $y = r + \max_a Q(s', a)$

Update the parameter  $\theta$  of the Q network so that  $Q(s, a)$  approximates the target y

Update state  $s = s'$ , step  $+ = 1$

When step is divisible by period C, update the target network  $Q' = Q$

End for

End for

---

## 4. Results

This section is the experimental part of the article, which first introduces the experimental environment and explains the method of generating data. Then, the experimental results are explained in detail, and a comprehensive comparison is made between the G-DQN algorithm and the other three methods to prove the effectiveness of the G-DQN algorithm.

### 4.1. Experimental Environment

The hardware environment in this paper is 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, the simulation environment is SUMO 1.14.1, the deep learning framework used is Keras, and the experiments are run in python 3.8.

The hyperparameter settings are shown in the following Table 2.

**Table 2.** Hyperparameters.

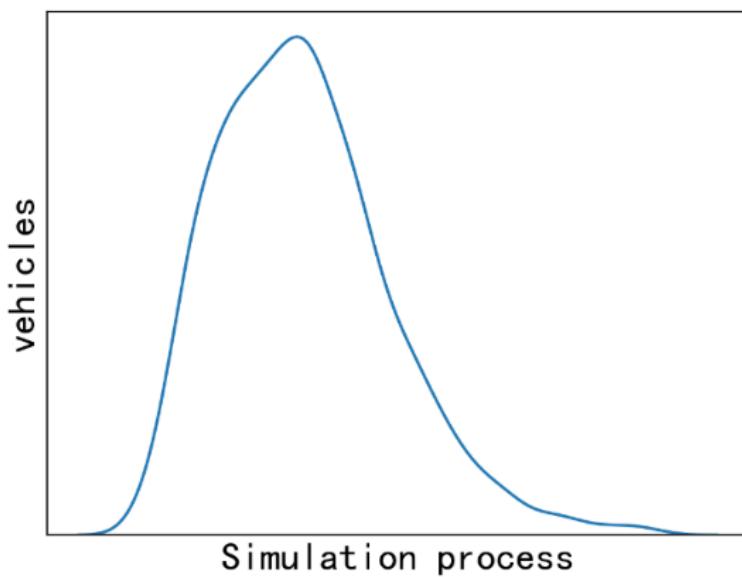
Hyperparameter	Value
Number of training rounds episodes	30
Number of training steps per round steps	5400
Batch size	20
Experience pool size M	50,000
Learning Rate $\alpha$	0.001
Discount Factor $\gamma$	0.75
Initial learning strategy probability $\epsilon$	1.0
Target network update cycle C	500
Green light duration	12
Yellow light duration	4
$k_1$	0.8
$k_2$	0.2

### 4.2. Experimental Data

The two parameters of the Weibull distribution, shape parameters and scale parameters, both have intuitive physical meanings. The shape parameter determines the shape of the distribution, while the scale parameter determines the scale of the distribution. This makes the analysis results easier to interpret and apply to practical problems. As shown in Figure 5, when the shape parameter of the Weibull distribution is 2, the Weibull distribution coincides with the peak hour traffic flow, showing a trend of increasing first and then decreasing [38]. Therefore, in order to make the experiment more realistic, we used the Weibull distribution in simulating vehicle generation. The Weibull distribution is a continuous probability distribution with probability density of [39]:

$$f(x; \lambda, k) = \begin{cases} \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (8)$$

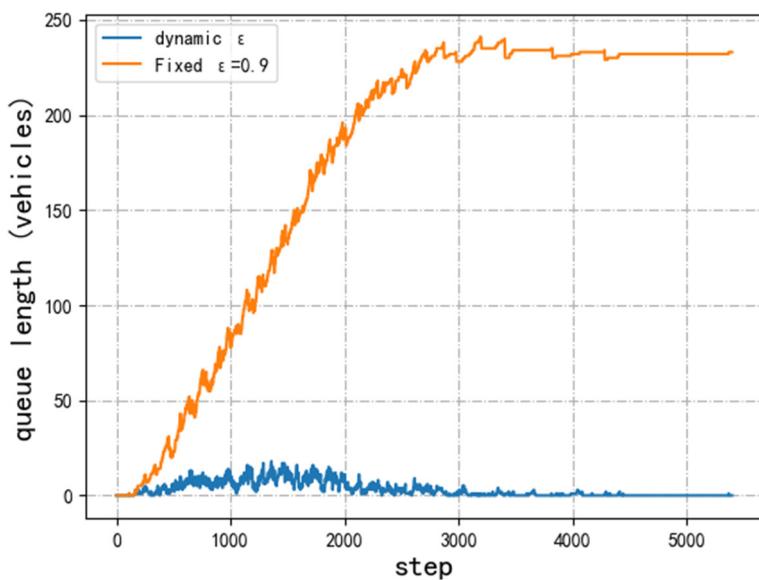
where  $x$  is the random variable,  $\lambda > 0$  is the scale parameter, and  $k > 0$  is the shape parameter. When  $k = 1$ , it is an exponential distribution; when  $k = 2$ , it is a Rayleigh distribution. The number of vehicles during the peak traffic hours showed a trend of rising and then falling, with the number of vehicles rising steadily at the beginning and gradually entering the peak hours; after the peak, the number of vehicles gradually decreased. Our experiments used the Traci interface in SUMO to associate SUMO with python, specifically defining a class for generating vehicles; the class initialized the total number of vehicles and the total number of steps in the experiment, and the class defined a function for generating vehicles, the function's function would randomly generate 1000 vehicles as the experimental data source; and each vehicle was set with a random initial place and destination.



**Figure 5.** Weibull distribution for shape parameter of 2.

#### 4.3. Experimental Results

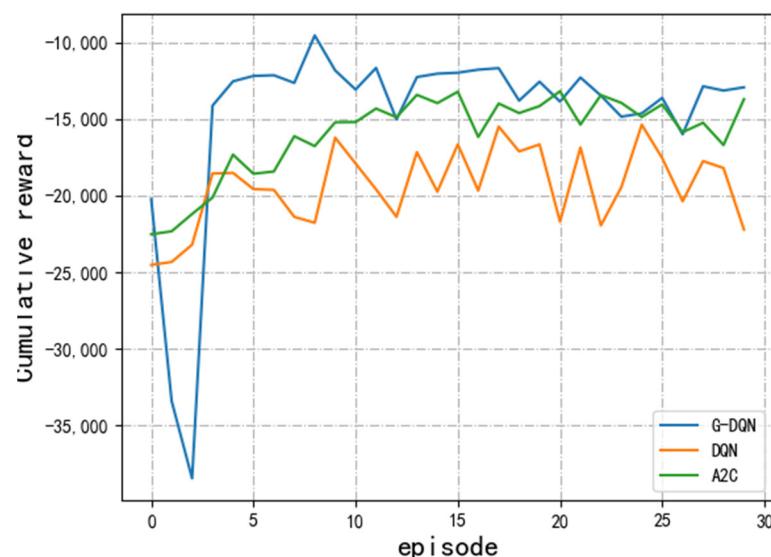
In this paper, we first experimented with a dynamic  $\epsilon$ -greedy action selection strategy. The experimental process made  $\epsilon$  dynamically adjustable, set to 1 at the beginning of training, and then afterwards it was each round minus the current number of rounds, divided by the total number of rounds. This avoided overfitting of the neural network and also sped up the convergence of the model. In order to verify the feasibility of this approach, a comparison experiment was set up in this paper, with a fixed  $\epsilon = 0.9$  action selection strategy according to the reference, and both experiments used the convolutional neural network DQN algorithm for model training and a final round of testing. The analysis was performed in terms of the metric of the number of vehicles in the queue, and the comparison of the two methods is shown in Figure 6. During testing, the number of vehicles queuing in the environment optimized by the dynamic  $\epsilon$ -greedy method was significantly less than a fixed  $\epsilon$ -Greedy method. This indicates the dynamic  $\epsilon$ -greedy method has a stronger traffic diversion ability and can maintain vehicles at intersections at a relatively low level.



**Figure 6.** Comparison of the number of queued vehicles for dynamic and static action selection strategies.

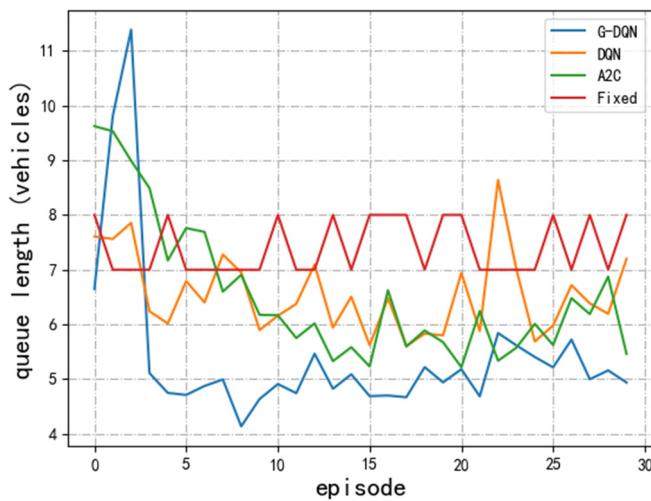
In summary, the dynamic  $\epsilon$ -greedy action selection strategy is significantly better than the static  $\epsilon$ -greedy action selection strategy. Therefore, we applied dynamic  $\epsilon$ -greedy to the original DQN algorithm and continued to optimize the algorithm. In this paper, we proposed a road division strategy based on subgroup series for obtaining traffic states. Since vehicles close to intersections have a greater degree of traffic impact and thus cause more traffic pressure, we proposed the G-DQN algorithm by dividing the divided intersection cells into groups. The larger the weight of the group closer to the intersection, the greater its impact on the traffic road condition was indicated. Compared with the average way of dividing roads, the method proposed in this paper can reflect the current traffic state more accurately, and at the same time can normalize the matrix, which is convenient for neural network training.

We conducted model training and testing on the G-DQN algorithm, DQN algorithm, A2C algorithm, and signal light fixed-timing scheme in the same environment, and compared and analyzed three indicators: cumulative reward, average waiting time, and average queue length. Among them, the cumulative reward reflects the speed of the rate of convergence of the neural network, which is used to display the quality of the traffic conditions laterally. Due to the lack of cumulative reward indicators in the fixed timing scheme of signal lights, the comparison of cumulative rewards among the three deep reinforcement algorithms is shown in Figure 7. At the beginning of the algorithm, the cumulative rewards of the G-DQN algorithm first declined to a certain extent, and the cumulative rewards of the DQN algorithm were more stable, while the A2C algorithm, as a new deep reinforcement learning algorithm, had a more stable state. With the training to about five episodes, the performance of the G-DQN algorithm was significantly better than that of the DQN algorithm, the G-DQN algorithm had a faster rate of convergence than DQN, and the performance of the final convergence was similar to that of the A2C algorithm.



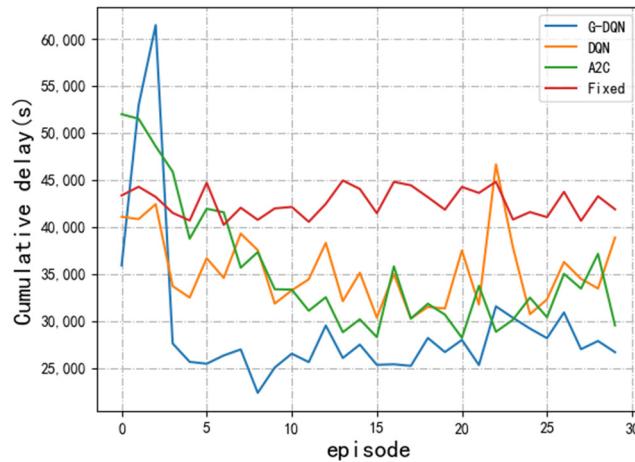
**Figure 7.** Comparison of the accumulated rewards of the three algorithms.

The average queue length comparison of the four methods is shown in Figure 8. At the beginning of training, the number of vehicle queues in the G-DQN algorithm environment was higher than the other three methods, and the stability of these three methods was better than that of the G-DQN algorithm at the beginning. However, after several rounds of training, the number of queued vehicles in the G-DQN algorithm environment sharply decreased. The number of queued vehicles in the environment was the lowest among the four methods, and the convergence trend appeared faster than in the DQN algorithm environment, reflecting the performance superiority of the G-DQN algorithm.



**Figure 8.** Comparison of the average length of the four methods.

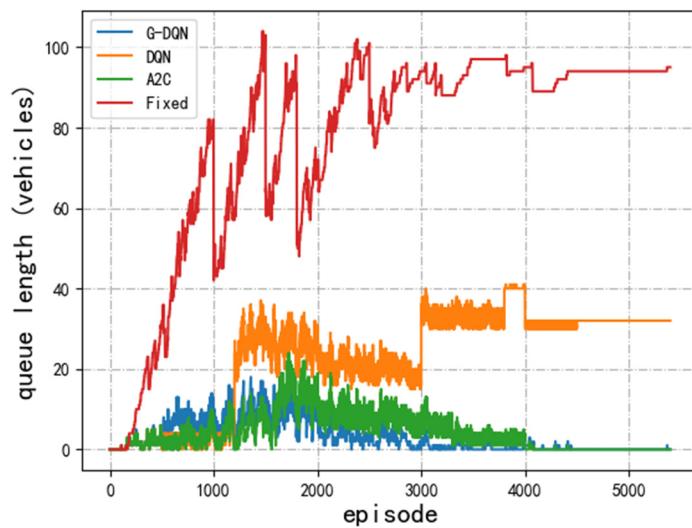
The cumulative queue time comparison of the four methods is shown in Figure 9. Similarly, at the beginning of the training, the three deep reinforcement learning algorithms did not converge, with large fluctuations, and the cumulative queue time was even higher than the signal fixed-timing scheme. However, after several rounds of training, the effect of the three deep reinforcement learning schemes was obviously better than that of the signal light fixed-timing scheme. Compared with the DQN algorithm, the cumulative waiting time under the G-DQN algorithm environment was less than that under the DQN algorithm environment, and the rate of convergence of the G-DQN algorithm is significantly faster than that of the DQN algorithm. Compared with A2C, although the G-DQN algorithm had lower stability, the vehicle queuing time in the environment was shorter than that of A2C algorithm.



**Figure 9.** Comparison of cumulative queuing time of the four methods.

Finally, we conducted a round of testing on four methods and analyzed and compared the number of queued vehicles under simulated road conditions lasting for 90 min. The experimental results are shown in Figure 10. At the beginning of the testing phase, the number of vehicles queuing in all four methods' environments increased, with the A2C algorithm environment having the least number of vehicles queuing. Under the environment of fixed-signal timing scheme, the number of vehicles queuing up was obviously more than the three deep reinforcement learning algorithms, which proves the feasibility of the deep reinforcement learning algorithm. And within 1200 steps of testing, the number of queued vehicles in the DQN algorithm and A2C algorithm environments was less than that in the G-DQN algorithm environment, which is consistent with the conclusion of the model

training process mentioned above. As the testing progressed, the stability of the G-DQN algorithm began to manifest. In the G-DQN algorithm environment, the increasing trend of the number of queued vehicles was slow, and the peak value of queued vehicles was smaller than that in the A2C algorithm environment. After 2000 steps, there was a clear downward trend, and the number of queued vehicles in the environment was almost zero when the algorithm converged. Due to the lack of improvement in the state and rewards of the DQN algorithm, it cannot effectively capture the features in the environment, which can have an impact on optimization. Therefore, there was a significant fluctuation in the number of queued vehicles in the DQN algorithm environment, which proves that the DQN algorithm is not very stable, and the number of queued vehicles when the algorithm converges is significantly higher than in the G-DQN algorithm. Although the number of queued vehicles in both the G-DQN and A2C algorithm environments tended to zero in the end, it can be seen that the number of queued vehicles decreased faster in the G-DQN algorithm environment. Therefore, the comprehensive performance of the G-DQN algorithm is better. However, the G-DQN algorithm started to have poor stability, which is an aspect to be optimized in the future.



**Figure 10.** Comparison of the number of vehicles in queue for the four methods.

In summary, as shown in Table 3, during the entire experiment, the fixed timing of the signal lights did not learn any traffic information, so the performance was poor. The other three reinforcement learning algorithms all apply better dynamics than the  $\epsilon$ -Greedy strategy to select actions. According to the cumulative reward comparison in Figure 7, all three deep reinforcement learning algorithms tended to converge during the training process, and the G-DQN and A2C algorithms converged more stably. According to the comparison of vehicle queue length and cumulative waiting time in Figures 8 and 9, G-DQN and A2C showed better performance during the training process, with both indicators maintained at lower levels. Therefore, in the final testing phase, the G-DQN and A2C algorithms were able to handle 1000 vehicles in the environment well, ensuring that there were no congested vehicles in the final testing phase of the environment. The difference lies in the G-DQN reflecting the state of the intersection in more detail, optimizing the details of the model, designing a more realistic state space, and providing more comprehensive rewards. Therefore, the neural network has faster convergence speed and better performance. The experimental results show that when the intersection enters peak hours, the model trained by the G-DQN algorithm can arrange the traffic lights in a more reasonable phase. This optimizes the traffic environment with signal lights, improving the operational efficiency and safety of road traffic.

**Table 3.** Comparison of experimental results.

	Accumulated Rewards	Average Queue Length (Vehicles)	Cumulative Queuing Time (s)	Number of Vehicle Queues Tested (Vehicles)	Neural Network Convergence Step during Testing
G-DQN	−12,935.5	4.9	26,676	0	3200
DQN	−22,209.0	7.2	38,870	32	4100
A2C	−13,709.3	5.5	29,524	0	3600
Fixed	−	8.0	40,538	95	−

## 5. Conclusions

The optimization and control of traffic signals is of great significance for improving logistics efficiency, improving distribution reliability, optimizing distribution networks, and improving the level of logistics informatization. It can promote the healthy development of the logistics industry. Deep reinforcement learning provides a method to optimize the signal optimization control system and provides a solution for urban traffic optimization control. In this paper, the DQN algorithm based on a convolutional neural network (CNN) model is used to model regional traffic, and then the concept of grouping sequence is introduced on the basis of the algorithm, and the G-DQN algorithm is proposed. We compared the optimization performance of the DQN algorithm, G-DQN algorithm, and A2C algorithm. Experiments have shown that the G-DQN algorithm has significant advantages in dealing with the optimization control of regional traffic lights during peak hours. After optimization, the number and time of vehicle queues in the area are significantly better than other methods, making it more suitable for dealing with the current traffic congestion problems in large cities. This work systematically studied the optimization control of regional traffic signal lights and achieved some research results with theoretical and practical application value. However, with the increase in urban vehicles, the number of high-volume traffic intersections will also increase day by day, and constructing a single intersection traffic model can no longer meet the needs of the intelligent transportation field. In the future, we will first consider optimizing traffic signal control in multi-intersection areas and regulating traffic signals in more complex environments by using more complex road network environments, such as irregular multiple intersections and intersections with more than four directions, and using more complex traffic environments, such as adding buses or sidewalks. Secondly, we will focus on improving more advanced deep reinforcement learning algorithms, such as A3C, Twin Delayed Deep Deterministic Policy Gradients (TD3) and Reinforcement Learning with Expert Demonstrations and Minimal Losses (REM). We will add convolutional layers to the neural networks in these algorithms for finer feature extraction, and consider adding residual networks to improve network performance. Finally, we will further adjust the definitions of status and rewards to provide a more detailed description of the transportation environment.

**Author Contributions:** Conceptualization, K.C. and L.W.; methodology, L.W.; software, L.W.; validation, S.Z. and L.D.; formal analysis, H.J.; investigation, H.J.; resources, K.C.; data curation, L.W.; writing—original draft preparation, L.W., G.J., S.S. and H.Z.; writing—review and editing, K.C.; visualization, L.W.; supervision, S.Z.; project administration, L.D.; funding acquisition, K.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the MSIT (Ministry of Science and ICT), Republic of Korea, under the Innovative Human Resource Development for Local Intellectualization support program (IITP-2024-RS-2022-00156334) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). We also received support from the 2023 Liaoning Provincial Education Department's Basic Research General Project (JYTM20231518).

**Data Availability Statement:** Data supporting this study are included within the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Min, Z. Enhance the management of roads in central to the development of the “two-oriented society” the significance of logistics. *Logist. Eng. Manag.* **2009**, *31*, 40–41.
- Naidoo, K. Using intelligent transport technologies in SA’s largest urban areas. *Civ. Eng. Mag. S. Afr. Inst. Civ. Eng.* **2016**, *2016*, 51–54.
- Jin, W.; Salek, M.S.; Chowdhury, M.; Torkjazi, M.; Huynh, N.; Gerard, P. Assessment of Operational Effectiveness of Synchro Green Adaptive Signal Control System in South Carolina. *Transp. Res. Rec.* **2021**, *2675*, 714–728. [[CrossRef](#)]
- Maipradit, A.; Kawakami, T.; Liu, Y.; Gao, J.; Ito, M. An Adaptive Traffic Signal Control Scheme Based on Back-Pressure with Global Information. *J. Inf. Process.* **2021**, *29*, 124–131. [[CrossRef](#)]
- Tai, T.S.; Yeung, C.H. Adaptive strategies for route selection en-route in transportation networks. *Chin. J. Phys.* **2022**, *77*, 712–720. [[CrossRef](#)]
- Cahyono, S.D.; Tristono, T.; Sutomo; Utomo, P. Model of demand order method of traffic lights phases. *J. Phys. Conf. Ser.* **2019**, *1211*, 012036. [[CrossRef](#)]
- Noaeen, M.; Naik, A.; Goodman, L.; Crebo, J.; Abrar, T.; Abad, Z.S.H.; Bazzan, A.L.; Far, B. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Syst. Appl.* **2022**, *199*, 116830. [[CrossRef](#)]
- Kuang, L.; Zheng, J.; Li, K.; Gao, H. Intelligent Traffic Signal Control Based on Reinforcement Learning with State Reduction for Smart Cities. *ACM Trans. Internet Technol. (TOIT)* **2021**, *21*, 102. [[CrossRef](#)]
- Müller, A.; Sabatelli, M. Safe and Psychologically Pleasant Traffic Signal Control with Reinforcement Learning Using Action Masking. In Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, China, 8–12 October 2022.
- Zhao, W.; Ye, Y.; Ding, J.; Wang, T.; Wei, T.; Chen, M. IPDALight: Intensity- and phase duration-aware traffic signal control based on Reinforcement Learning. *J. Syst. Archit.* **2022**, *123*, 102374. [[CrossRef](#)]
- Genders, W.; Razavi, S.; Asce, A.M. Policy Analysis of Adaptive Traffic Signal Control Using Reinforcement Learning. *J. Comput. Civ. Eng.* **2019**, *34*, 04019046. [[CrossRef](#)]
- Zhang, R.; Ishikawa, A.; Wang, W.; Striner, B.; Tonguz, O.K. Using Reinforcement Learning with Partial Vehicle Detection for Intelligent Traffic Signal Control. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 404–415. [[CrossRef](#)]
- Boukerche, A.; Zhong, D.; Sun, P. A Novel Reinforcement Learning-Based Cooperative Traffic Signal System through Max-pressure Control. *IEEE Trans. Veh. Technol.* **2021**, *71*, 1187–1198. [[CrossRef](#)]
- Alegre, L.N.; Ziemke, T.; Bazzan, A.L.C. Using Reinforcement Learning to Control Traffic Signals in a Real-World Scenario: An Approach Based on Linear Function Approximation. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 9126–9135. [[CrossRef](#)]
- Wang, X.; Ke, L.; Qiao, Z.; Chai, X. Large-Scale Traffic Signal Control Using a Novel Multiagent Reinforcement Learning. *IEEE Trans. Cybern.* **2020**, *51*, 174–187. [[CrossRef](#)] [[PubMed](#)]
- Antes, T.D.O.; Bazzan, A.L.C.; Tavares, A.R. Information upwards, recommendation downwards: Reinforcement learning with hierarchy for traffic signal control. *Procedia Comput. Sci.* **2022**, *201*, 24–31. [[CrossRef](#)]
- Li, H.; Kumar, N.; Chen, R.; Georgiou, P. A deep reinforcement learning framework for Identifying funny scenes in movies. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018.
- Park, S.; Han, E.; Park, S.; Jeong, H.; Yun, I. Deep Q-network-based traffic signal control models. *PLoS ONE* **2021**, *16*, e0256405. [[CrossRef](#)] [[PubMed](#)]
- Zhao, T.; Wang, P.; Li, S. Traffic Signal Control with Deep Reinforcement Learning. In Proceedings of the 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS), Madurai, India, 15–17 May 2019.
- Wan, C.H.; Hwang, M.C. Adaptive Traffic Signal Control Methods Based on Deep Reinforcement Learning. In *Intelligent Transport Systems for Everyone’s Mobility*; Springer: Singapore, 2019; pp. 195–209.
- Zou, Y.; Qin, Z. Value-Based Bayesian Meta-Reinforcement Learning and Traffic Signal Control. *arXiv* **2020**, arXiv:2010.00163.
- Chu, K.F.; Lam, A.Y.S.; Li, V.O.K. Traffic Signal Control Using End-to-End Off-Policy Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 7184–7195. [[CrossRef](#)]
- Muresan, M.; Pan, G.; Fu, L. Multi-Intersection Control with Deep Reinforcement Learning and Ring-and-Barrier Controllers. *Transp. Res. Rec.* **2021**, *2675*, 308–319. [[CrossRef](#)]
- Alhassan, A.; Saeed, M. Adjusting Street Plans Using Deep Reinforcement Learning. In Proceedings of the 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 26 February–1 March 2021.
- Bouktif, S.; Cheniki, A.; Ouni, A.; El-Sayed, H. Deep reinforcement learning for traffic signal control with consistent state and reward design approach. *Knowl. Based Syst.* **2023**, *267*, 110440. [[CrossRef](#)]
- Su, H.; Zhong, Y.D.; Chow, J.Y.; Dey, B.; Jin, L. EMVLight: A multi-agent reinforcement learning framework for an emergency vehicle decentralized routing and traffic signal control system. *Transp. Res. Part C Emerg. Technol.* **2023**, *146*, 103955. [[CrossRef](#)]
- Ramos-Martinez, M.; Torres-Cantero, C.A.; Ortiz-Torres, G.; Sorcia-Vázquez, F.D.; Avila-George, H.; Lozoya-Ponce, R.E.; Vargas-Méndez, R.A.; Renteria-Vargas, E.M.; Rumbo-Morales, J.Y. Control for Bioethanol Production in a Pressure Swing Adsorption Process Using an Artificial Neural Network. *Mathematics* **2023**, *11*, 3967. [[CrossRef](#)]

28. Rentería-Vargas, E.M.; Aguilar, C.J.Z.; Morales, J.Y.R.; De-La-Torre, M.; Cervantes, J.A.; Huerta, J.R.L.; Torres, G.O.; Vázquez, F.D.J.S.; Sánchez, R.O. Identification by Recurrent Neural Networks applied to a Pressure Swing Adsorption Process for Ethanol Purification. In Proceedings of the 2022 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 21–22 September 2022; pp. 128–134.
29. Behrisch, M.; Bieker, L.; Erdmann, J.; Krajzewicz, D. SUMO—Simulation of Urban MObility: An Overview. In Proceedings of the SIMUL 2011, the Third International Conference on Advances in System Simulation, Barcelona, Spain, 23–28 October 2011.
30. Özdiç, A.; Yilmaz, B. Medical infrared thermal image based fatty liver classification using machine and deep learning. *Quant. InfraRed Thermogr. J.* **2023**. [[CrossRef](#)]
31. Garrido, I.; Lagüela, S.; Fang, Q.; Arias, P. Introduction of the combination of thermal fundamentals and Deep Learning for the automatic thermographic inspection of thermal bridges and water-related problems in infrastructures. *Quant. InfraRed Thermogr. J.* **2022**, *20*, 231–255. [[CrossRef](#)]
32. Chebbah, N.K.; Ouslim, M.; Benabid, S. New computer aided diagnostic system using deep neural network and SVM to detect breast cancer in thermography. *Quant. InfraRed Thermogr. J.* **2023**, *20*, 62–77. [[CrossRef](#)]
33. Mahoro, E.; Akhloufi, M.A. Breast cancer classification on thermograms using deep CNN and transformers. *Quant. InfraRed Thermogr. J.* **2022**. [[CrossRef](#)]
34. Ervural, S.; Ceylan, M. Thermogram classification using deep siamese network for neonatal disease detection with limited data. *Quant. InfraRed Thermogr. J.* **2022**, *19*, 312–330. [[CrossRef](#)]
35. Barto, A.G.; Sutton, R.S. Reinforcement learning: An introduction (Adaptive computation and machine learning). *IEEE Trans. Neural Netw.* **1998**, *9*, 1054.
36. Kanis, S.; Samson, L.; Bloembergen, D.; Bakker, T. Back to Basics: Deep Reinforcement Learning in Traffic Signal Control. *arXiv* **2021**, arXiv:2109.07180.
37. Xu, Y.; Ni, Z. Intelligent traffic signal coordination control method based on YOLOv3 and DQN. In Proceedings of the International Conference on Signal Processing and Communication Technology (SPCT 2022), Harbin, China, 23–25 December 2022.
38. Modarres, M.; Kaminskiy, M.P.; Krivtsov, V. *Reliability Engineering and Risk Analysis: A Practical Guide*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2009.
39. Johnson, N.L.; Kotz, S.; Balakrishnan, N. *Continuous Univariate Distributions*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 1994; Volume 1.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.