# Graph Convolution-Based Deep Reinforcement Learning for Multi-Agent Decision-Making in Interactive Traffic Scenarios

Liu, Qi; Li, Zirui ; Li, Xueyuan ; Wu, Jingda ; Yuan, Shihua

**Citation (APA)**
Liu, Q., Li, Z., Li, X., Wu, J., & Yuan, S. (2022). Graph Convolution-Based Deep Reinforcement Learning for Multi-Agent Decision-Making in Interactive Traffic Scenarios. In *Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 4074-4081). IEEE. https://doi.org/10.1109/ITSC55140.2022.9922001

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

# Graph Convolution-Based Deep Reinforcement Learning for Multi-Agent Decision-Making in Interactive Traffic Scenarios

Qi Liu*, Zirui Li*, Xueyuan Li, Jingda Wu, Shihua Yuan

*Abstract*— A reliable multi-agent decision-making system is highly demanded for safe and efficient operations of connected and autonomous vehicles (CAVs). In order to represent the mutual effects between vehicles and model the dynamic traffic environments, this research proposes an integrated and open-source framework to realize different Graph Reinforcement Learning (GRL) methods for better decision-making in interactive driving scenarios. Firstly, an interactive driving scenario on the highway with two ramps is constructed. The vehicles in this scenario are modeled by graph representation, and features are extracted via Graph Neural Network (GNN). Secondly, several GRL approaches are implemented and compared in detail. Finally, The simulation in the SUMO platform is carried out to evaluate the performance of different GRL approaches. Results are analyzed from multiple perspectives to compare the performance of different GRL methods in intelligent transportation scenarios. Experiments show that the implementation of GNN can well model the interactions between vehicles, and the proposed framework can improve the overall performance of multi-agent decision-making. The source code of our work can be found at **https://github.com/Jacklinkk/TorchGRL**.

## I. INTRODUCTION

Autonomous vehicles operate in highly complex driving environments. An interactive traffic scenario refers to a driving environment that is highly cooperative and dynamic. Decision-making for autonomous vehicles should consider the interaction with surrounding vehicles, and the mutual effect between different traffic participants has a great influence on decision-making [1]. It is critically important for each autonomous vehicle in interactive traffic scenarios to generate appropriate and cooperative behaviors. Therefore, intelligent multi-agent decision-making technology is highly demanded for autonomous vehicles to efficiently handle complex road environments and multi-agent interactions [2].

The detailed overviews of decision-making technology for autonomous vehicles are presented in [3], [4]. In general, current technologies mainly focus on reinforcement learning (RL) methods due to the high complexity of the driving environments and the frequency of interaction with different agents [5]. The keys of RL in interactive traffic scenarios can

be summarized in the two following aspects: 1) The accurate representation of the feature of each agent and the interaction between different vehicles. 2) The efficient modeling of the driving policy to generate reasonable behaviors [6]. Considering that the RL methods with a manually designed policy model are easily affected by prior knowledge, which suffers from several weaknesses, including accuracy of generated behavior in interactive environments and generality between different scenarios. With the rapid development of deep learning methods in supervised learning, incorporating neural networks into RL frameworks has shown a large potential to improve the performance of RL methods in recent research. Thus, deep reinforcement learning (DRL) methods have been proposed to ensure better high efficient exploration and learning process of each agent [7].

There have been some studies on the decision-making in interactive traffic scenarios using DRL methods [8]– [13]. However, such approaches only consider the individual features of each vehicle as the input of DRL, ignoring the mutual effects between pairs of vehicles. This will generate low cooperative behaviors in interactive traffic scenarios, which may lead to danger or even traffic accidents. Graph representation can accurately describe the mutual effects between pairs of agents, which enables modeling the relationship of vehicles in interactive traffic scenarios. Thus, some researchers focus on GRL methods to model the interaction with graph representation.

GRL methods combine GNN and DRL to achieve better decision generation in interactive scenarios. The features of interactive scenarios are proceeded by GNN, and cooperative behaviors are generated by DRL framework. GRL methods have great application prospects in intelligent transportation systems; for instance, traffic light control [14], vehicle routing [15], trajectory prediction [16], [17] and vehicle repositioning [18]. There are also some related works in multi-agent decision-making based on the GRL approach. In [19], graph convolutional reinforcement learning method is proposed for multi-agent decision-making; two multi-head attention graph convolutional layers are utilized for features extraction of agents, and behaviors for different agents are generated by DQN. The experiment is carried out on the "MAgent" platform, and results show that the proposed methods substantially outperform existing methods in various of cooperative scenarios. In [20], GNN and DQN are combined (named GCQ) for multi-agent cooperative controlling of CAVs in merging scenarios. Specifically, GNN is proposed to aggregate the information acquired from collaborative sensing, while cooperative lane-changing decisions are

*Qi Liu and Zirui Li contributed equally to this paper

Qi Liu, Zirui Li, Xueyuan Li, and Shihua Yuan are with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China. (E-mails: 3120195257@bit.edu.cn; z.li@bit.edu.cn; lixueyuan@bit.edu.cn; yuanshihua@bit.edu.cn.)

Zirui Li is also with Department of Transport and Planning, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Stevinweg 1, 2628 CN Delft, The Netherlands.

Jingda Wu is with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore, 639798. (E-mail: jingda001@e.ntu.edu.sg)

(Corresponding author: Xueyuan Li)

generated from DQN. The experiment is carried out in the SUMO platform, results show good performance compared with the rule-based method and LSTM-based Q-learning method. A similar research is put forward in [21], the main difference is that a reward function matrix is proposed for training various decision-making modes to achieve a better trade-off between safety and efficiency.

This paper proposes an innovative modular framework to analyze the performance of different GRL methods in interactive traffic scenarios. The traffic scenario adopted in our work is constructed based on [20]. The designed algorithm in this paper is based on GRL, which consists of two modules: GNN and DRL. Specifically, several Deep Q-Learning methods are utilized in the DRL module, including DQN, Double DQN, Dueling DQN, and Dueling-Double DQN (D3QN). The graph representation of the multi-agent, including node feature matrix, adjacency matrix, and RL-filter, are selected as inputs, while the behaviors of each agent are generated as output. The general framework of our approach is modularized and developed based on python, different categories of GNN and DRL algorithms can be replaced according to actual needs. Finally, simulation is carried out based on the SUMO platform [22], and the results are discussed in detail from multiple perspectives and dimensions. The schematic diagram of the designed framework is shown in Fig.1.

The main contributions of this research are summarized as follows:

(1) Compared with [19], we applied the GRL method to multi-agent decision-making for highway ramping of autonomous vehicles. The constructed traffic scenario is dynamic, and the number of agents varies with time.

(2) Compared with [20], we propose a general GRL framework to solve the decision-making issues in interactive traffic scenarios. Various GRL models under the proposed framework are also compared and analyzed.

(3) An open-source library with detailed instructions is constructed. This library enables the verification of different GRL methods; it can also be extended to various types of interactive traffic scenarios.

## II. PROBLEM FORMULATION

The purpose of our work is to generate cooperative lane-change decisions for autonomous vehicles in an interactive traffic scenario. A framework implemented with several GRL algorithms is proposed, which consists of three key components: Scenario Construction, GRL Algorithm, and Simulation. The vehicles in the scenario are modeled as a graph, where nodes represent different vehicles and edges represent the interplay between every two vehicles. Graph representation is then processed into node features, adjacency matrix, and RL-filter. The GRL Algorithm is developed to train lane-changing policy, which takes graph representation as input and generates the Q values [23] of different lane-change actions. The simulation part takes Q values as input, and motions of each vehicle are generated to update the

scenario, thus enabling the continuous training of the GRL framework.

The scenario in our research is constructed based on [20], which is shown in Fig.2. The scenario includes a 3-lane highway with two ramp exits. In this scenario, two types of vehicles operate: human-driven vehicles (HVs) and autonomous vehicles (AVs). Different vehicles have different driving tasks, and they need to cooperate to complete the scheduled driving tasks more efficiently. White vehicles represent HVs, entering from the left side of the highway and exiting from the right side. Colored vehicles represent AVs and enter from the left side of the freeway, specifically red vehicles exiting at ramp 1 and green vehicles exiting at ramp 2.

AVs interact with the traffic environment by taking lane-change actions at discrete time steps in the constructed traffic scenario. Upon taking the actions, the state of the traffic environment changes, and AVs receive a reward. Nevertheless, the environmental information can be completely observed within the observation range of AVs. Thus, this process can be modeled as Markov Decision Process (MDPs). For this constructed traffic scenario, four components define a finite horizon MDP: state space $(S)$, action apace $(a)$, reward function (R), and the discount factor $(\gamma)$ [24].

### A. State Representation

The scenario is modeled as an undirected graph. Each vehicle in this scenario is regarded as the node of the graph, and the interaction between vehicles is regarded as the edge of the graph. More formally, the constructed graph is described as $G = \{N, E\}$, where $N = \{n_i, i \in \{1, 2, ...n\}\}$ is a set of node attributes and $E = \{e_{ij}, i, j \in \{1, 2, ...n\}\}$ is a set of edge attributes. Specifically, $n$ denotes the number of nodes in the constructed graph that is equal to the total number of vehicles. Then, graph representation is utilized to model the characteristics and interaction of vehicles. The state space is discrete and can be described as follows:

$$S_t = [N_t, A_t, F_t] \tag{1}$$

where $N_t \in \mathbf{R}^{n \times f}$ denotes the node features matrix, $A_t \in \mathbf{R}^{n \times n}$ denotes the adjacency matrix and $F_t \in \mathbf{R}^{1 \times n}$ denotes the RL-filter matrix. The above three matrices together constitute the state space matrix, each of the matrix is manipulated as follows.

*1) Node Features Matrix:* Node features matrix represents the features of the constructed scenario. It consists of the feature matrix of each vehicle, which can be described as follows:

$$N_t = \begin{bmatrix} [V_1, X_1, L_1, I_1] \\ [V_2, X_2, L_2, I_2] \\ \cdots \\ [V_i, X_i, L_i, I_i] \\ \cdots \\ [V_n, X_n, L_n, I_n] \end{bmatrix} \tag{2}$$

where $[V_i, X_i, L_i, I_i]$ represents the normalized state matrix of each vehicle. Specifically in the lane-change task, $V_i =$
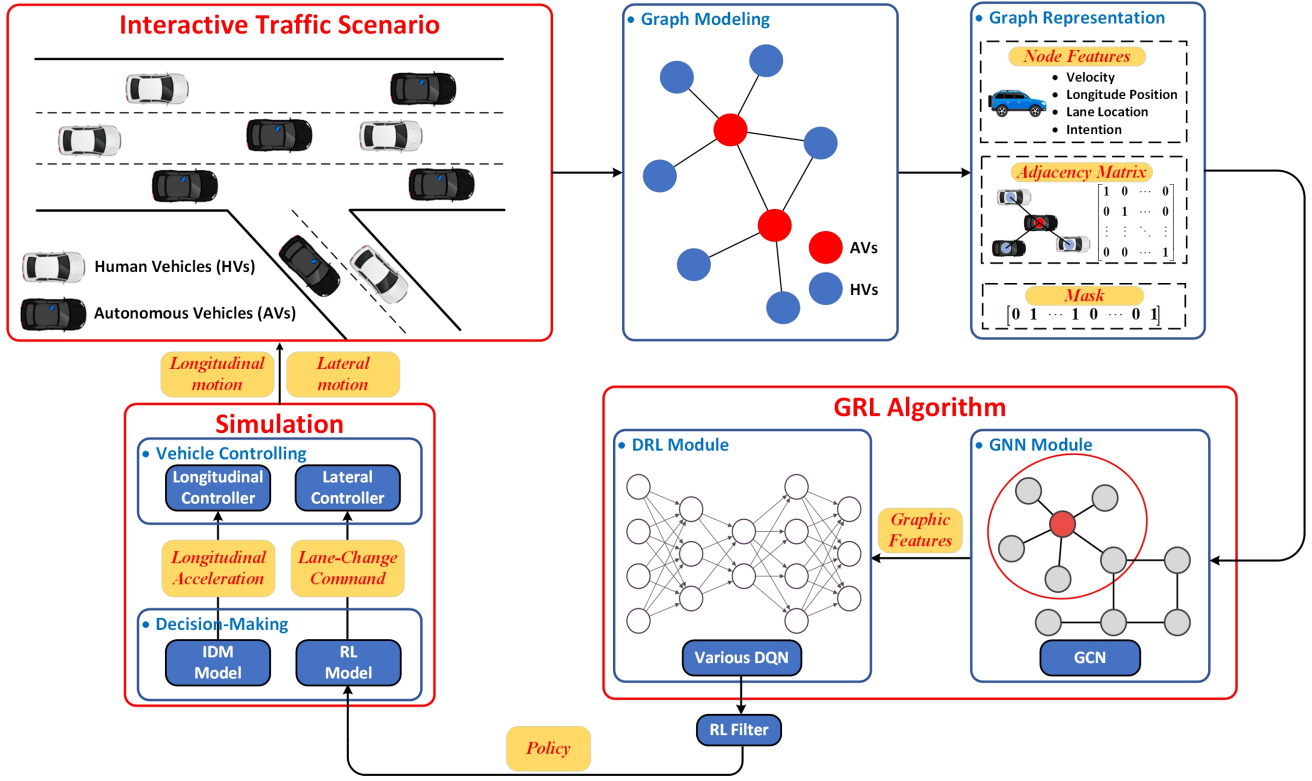
**4075**

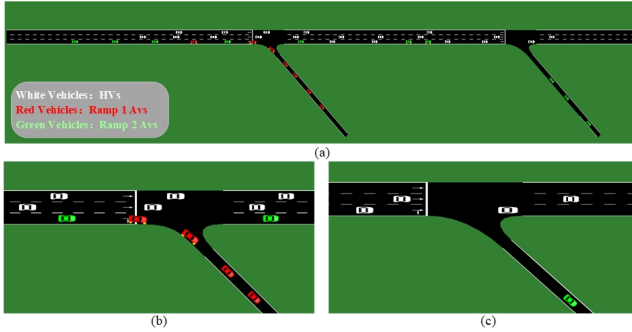Fig. 1. The schematic diagram of the proposed framework.



Fig. 2. The traffic scenario constructed in the proposed research. (a) is the whole view of the traffic scenario; (b) is the view of ramp 1 where red vehicles drive out; (c) is the view of ramp 2 where green vehicles drive out.

$V_{i\_actual}/V_{max}$ denotes the normalized longitudinal speed of vehicles relative to the maximum longitudinal speed; $X_i = X_{i\_longitudinal}/L_{highway}$ denotes the normalized longitudinal coordinate of vehicles relative to the length of highway; $L_i$ denotes the one-hot encoding matrix of current lane position (left-lane, right-lane and middle-lane) of vehicles; $I_i$ denotes the one-hot encoding matrix of current intention of (change to left-lane, change to right-lane and go straight) of vehicles.

*2) Adjacency Matrix:* Adjacency matrix represents the interaction between vehicles, which is embodied in the information sharing between vehicles in the constructed scenario. The adjacency matrix is calculated based on the following assumption: 1) All vehicles can share information

with themselves. 2) All AVs can share information. 3) All AVs can share information with HVs in their sensing range. The derivation of the adjacency matrix is as follows:

$$
A_t = \begin{bmatrix}
e_{11} & e_{12} & \cdots & & \cdots & e_{1n} \\
e_{21} & e_{22} & \cdots & & \cdots & e_{2n} \\
\vdots & \vdots & \ddots & & & \vdots \\
& & & e_{ij} & & \\
\vdots & \vdots & & & \ddots & \vdots \\
e_{n1} & e_{n2} & \cdots & & \cdots & e_{nn}
\end{bmatrix} \quad (3)
$$

where $e_{ij}$ denotes the edge value of $vehicle_i$ and $vehicle_j$. $e_{ij} = 1$ when $vehicle_i$ and $vehicle_j$ share information; while $e_{ij} = 0$ when $vehicle_i$ and $vehicle_j$ share no information.

*3) RL-filter Matrix:* The construction of the RL-filter matrix is to filter out the corresponding terms of HVs for the output of the GRL algorithm. The derivation of the RL-filter matrix is as follows:

$$
F_t = [f_1, f_2, \cdots, f_i, \cdots, f_n] \quad (4)
$$

where $f_i = 0$ or $1$. If $vehicle_i$ belongs to AVs controlled by GRL algorithm, $f_i = 1$; otherwise $f_i = 0$.

*B. Action Space*

The action space is discrete. At each time step, the action space of AVs in the scenario is composed of different lane changing instructions that can be described as follows:

$$
a = [change\ to\ left, go\ straight, change\ to\ right] \quad (5)
$$

## C. Reward Function

The details of the reward function can refer to [20]. The reward function consists of four parts: intention reward, average speed reward, lane-changing penalty, and collision penalty. The reward function is defined as follows:

$$R = w_1 R_I + w_2 R_{AS} + w_3 P_{LC} + w_4 P_C \quad (6)$$

where $w_i$ denotes the weight of each item; $R_I$ denotes the intention reward, $R_{AS}$ denotes the average speed reward; $P_{LC}$ denotes the lane-changing penalty, $P_C$ denotes the collision penalty.

## III. GRL ALGORITHM

The GRL algorithm is the critical part of the proposed framework, consisting of GNN and DRL. Based on the graph representation of scenario and reward functions, GRL can explore optimized lane-change strategies to ensure autonomous vehicles' efficient and safe driving task completion. The detailed algorithm design of the GNN and DRL module will be further illustrated. The detailed network structure of GRL is shown in Fig.3.

### A. GNN Module

The GNN module consists of fully connected layers (FC) and graph convolutional network (GCN). GCN is a type of GNN that extracts features of graph-structured data based on an efficient variant of convolutional neural networks [25]. Before implementing the graph convolution process, the node features matrix $N_t$ is firstly input into the FC to map the node features to sample markup space, which is described as follows:

$$N_{FC} = \phi^{FC}(N_t) \quad (7)$$

where $N_{FC}$ denotes the node feature matrix proceed by the FC; $\phi^{FC}$ denotes the neural network with FC.

Then, $N_{FC}$ and adjacency matrix $A_t$ are firstly input into the GCN to generate the graphical convolution feature, and the graph convolution process can be expressed by the following formulation [25]:

$$G_t = \phi^{GCN}(N_{FC}, A_t) = \sigma(D_t^{\frac{1}{2}} A_t D_t^{-\frac{1}{2}} N_{FC} W_t + b) \quad (8)$$

where $G_t$ denotes the graph convolutional features proceed from GCN; $\phi^{GCN}$ denotes the graph convolution operator; matrix $D_t$ is computed based on $A_t$, specifically, $D_{ii} = \Sigma_j A_{ij}$; $W_t$ denotes a layer-specific trainable weight matrix; $b$ denotes the offset, specifically, $b = 0$ is defined in this research; $\sigma$ denotes an activation function, specifically, ReLU [26] is chosen in this research.

Graph convolutional features are then input into the DRL module, then the Q value of lane changing actions for AVs is generated by filtering through the RL-filter matrix. The derivation of the Q value is as follows:

$$Q(s, a) = F_t \cdot [\phi^{DRL}(G_t)] \quad (9)$$

where $Q(s, a)$ denotes the Q value of lane-changing actions for AVs; $\phi^{DRL}$ denotes the policy neural network.

## B. DRL Module

This part elaborates on the principles and implementation process of different DRL algorithms carried out in this research.

*1) Deep Q-Network (DQN):* Q-learning is one of the most representative value-based RL methods based on the estimation of Q value [23]. Q value can be expressed as $Q(s, a)$, defined as the expectation of future rewards when taking action $\{a, a \in A\}$ at state $\{s, s \in S\}$ at a specific time step; in addition, reward $r$ is generated to the agent according to the response of the environment. The main idea of Q-learning is selecting the action that can obtain the greatest reward based on the Q value; then, the Q value is updated as follows:

$$Y_t^Q = R_{t+1} + \gamma \operatorname*{argmax}_a Q(s_{t+1}, a) \quad (10)$$

where $\gamma$ denotes the attenuation factor of the RL process.

However, Q-learning suffers from the curse of dimension. When the dimension of the state space is too large, the construction, searching, and updating of the Q value will cause high computational complexity; thus, the algorithm's efficiency will be greatly reduced or even become impossible to solve the RL problem. With the development of deep learning technology, DQN was proposed in [27], a neural network is constructed to estimate the Q value of the action space to extend the state space to high dimensions. The online Q network can be parametrized as $Q(s, a; \theta)$. The state space of the environment is chosen as the input of the neural network, while the Q value of the action space is generated as the output of the neural network. Finally the next action of the agent is determined.

DQN includes the following elements: replay buffer $D$ and a fixed target Q network parametrized as $Q(s, a; \theta')$. During the learning process of the agent, transition $(s, a, r, s')$ is stored in the replay buffer and sampled at each learning process. The calculation process of target Q value can be expressed by the following formula:

$$Y_t^{DQN} = R_{t+1} + \gamma \operatorname*{argmax}_a Q(s_{t+1}, a; \theta') \quad (11)$$

The output value of the target Q network $y$ is compared with the estimated value of the online Q network to calculate the loss, a gradient descent step on $(Y_i^{DQN} - Q(s_i, a_i; \theta))^2$ is carried out to update $\theta$. In addition, $\theta$ is synchronized to $\theta'$ every certain number of iterations.

DQN also suffers from several weaknesses: 1) The target Q value is overestimated due to the max operator. 2) The online Q network has the same structure as the target Q network, resulting in inefficient learning and convergence.

*2) Double DQN:* Double DQN is proposed to solve the overestimation problem of DQN [28]. Double DQN and DQN lie in the different estimation methods of the target Q value. The main idea is to construct different action selection and value evaluation functions. When updating the target Q value, the subsequent actions are selected from the online Q network, while the assessment of the Q value depends on
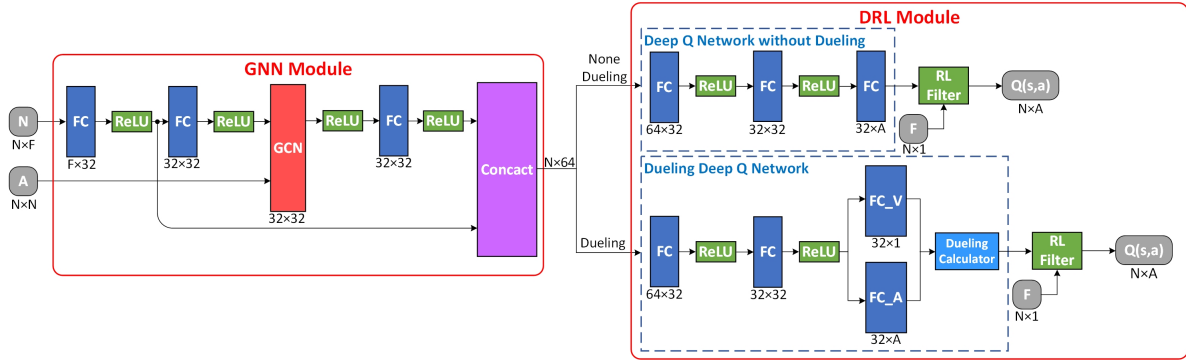
**4077**

Fig. 3. The network structures of GRL algorithm. N denotes the number of agents, F denotes the length of feature vector for each agent, A denotes the length of action space.

the target Q network. The estimation of the target Q value of Double DQN is derived as follows:

$$Y_t^{\text{Double}} = R_{t+1} + \gamma \; Q(s_{t+1}, \underset{a}{\arg\max} Q(s_{t+1}, a; \theta); \theta') \quad (12)$$

*3) Dueling DQN:* Dueling DQN is proposed to optimize the network structures of DQN [29]. Specifically, Dueling DQN consists of two separate estimators: one for the state value function, defined as $V(s; \theta, \beta)$; another for the state-dependent action advantage function, defined as $A(s, a; \theta, \alpha)$. Outputs from the two estimators are finally combined to generate the target Q value $Q(s, a; \theta, \alpha, \beta)$. The main benefit of this factoring is to generalize learning across actions without imposing any change to the underlying RL algorithm to ensure better policy evaluation. The estimation of the Q function of Dueling DQN is derived as follows:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'; \theta, \alpha)) \quad (13)$$

where $\theta$ denotes the parameters of the online Q network, while $\alpha$ and $\beta$ denotes the parameters of the two streams of fully-connected layers; $|\mathcal{A}|$ denotes the absolute value of the average of $A(s, a; \theta, \alpha)$

Thus, the target Q value of Dueling DQN can be calculated as follows:

$$Y_t^{\text{Dueling}} = R_{t+1} + \gamma \; \underset{a}{\arg\max} Q(s_{t+1}, a; \theta', \alpha', \beta') \quad (14)$$

*4) Dueling Double DQN (D3QN):* D3QN combines the improvements from Double DQN and Dueling DQN to optimize the estimation of Q value while ensuring better strategy evaluation. The estimation of the target Q value is derived as follows:

$$Y_t^{\text{D3QN}} = R_{t+1} + \gamma \; Q(s_{t+1}, \underset{a}{\arg\max} Q(s_{t+1}, a; \theta, \alpha, \beta); \theta', \alpha', \beta') \quad (15)$$

## IV. EXPERIMENT

In this section, the construction of the simulation environment is described, and implementation details of the proposed research is presented.

### A. Simulation Environment

The proposed framework is developed based on python and various third-party libraries. The program structure is shown in Fig.4, and the information of implemented third-party libraries is presented in Table I.

TABLE I
THE INFORMATION OF IMPLEMENTED THIRD-PARTY LIBRARIES

| Library | Refs | Function |
|---|---|---|
| Pytorch | [30] | The core framework of neural network |
| Pfrl | [31] | The DRL library based on Pytorch |
| Pytorch Geometric | [32] | The GNN library based on Pytorch |
| FLOW Library | [33] | The interface between traffic simulators and DRL libraries |

The traffic scenario and experiments are constructed based on the SUMO platform. It consists of a lateral control module and longitudinal control module. Q value of different lane-change actions from the GRL algorithm is input into the lateral control module, and control commands are finally generated to update the status of each vehicle, thus enabling the continuous training of the GRL algorithm. The two control modules are described in detail as follows:

*1) Longitudinal Control Module:* Acceleration commands are generated from the longitudinal control module. The longitudinal control of both HVs and AVs is achieved by the IDM model [34] embedded in SUMO.

*2) Lateral Control Module:* Lane-change commands are generated from the lateral control module. The lateral control of HVs is achieved by the LC2013 lane-changing model [35] embedded in SUMO, while the lateral control of AVs is achieved by the proposed GRL algorithm.

### B. Implementation Details

Simulation parameters include the parameters of traffic scenario that are summarized in Table II, and the parameters of GRL algorithm that are summarized in Table III.

Different exploration strategies are adopted at different training stages. Define $\pi(s)$ as the lane-changing strategy adopted by the AVs in state $s$. In the beginning, a random
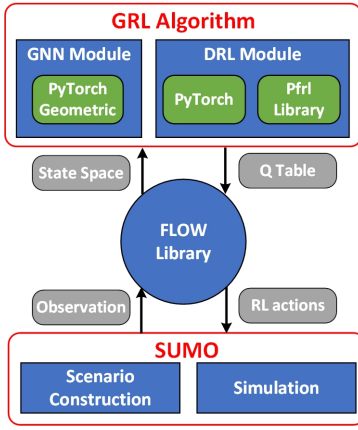
Fig. 4. The program structure of the proposed framework

TABLE II

PARAMETERS OF TRAFFIC SCENARIO

| Parameters | Symbols | Value |
|---|---|---|
| Number of HVs | $N_{\text{HVs}}$ | 20 |
| Number of AVs | $N_{\text{AVs}}$ | 20 |
| Number of AVs driven out by ramp 1 | $N_{\text{r1}}$ | 10 |
| Number of AVs driven out by ramp 2 | $N_{\text{r2}}$ | 10 |
| Length of Highway | $L$ | 500m |
| Longitudinal position of ramp 1 | $X_{\text{r1}}$ | 200m |
| Longitudinal position of ramp 2 | $X_{\text{r2}}$ | 400m |
| Speed limit for HVs | $V_{\text{maxHVs}}$ | 60km/h |
| Speed limit for AVs | $V_{\text{maxAVs}}$ | 75km/h |
| Inflow of HVs | $P_{\text{HVs}}$ | 0.3veh/s |
| Inflow of AVs | $P_{\text{AVs}}$ | 0.15veh/s |

exploration phase with a specific step is carried out to expand the exploration space of AVs. During the random exploration phase, AVs are sampled in the action space to generate random lane-change behavior, defined as $\pi(s) = \text{random}(a)$. After the random exploration phase, the epsilon-greedy strategy is used to generate the lane-change behavior of the AVs according to the Q table calculated from the GRL algorithm, and network parameters are updated at regular step intervals. The derivation of the epsilon-greedy strategy is as follows:

$$\pi(s) = \begin{cases} \text{random}(a) & P = \epsilon \\ \text{argmax}Q(s_t, a) & P = 1 - \epsilon \end{cases} \quad (16)$$

During the testing process, the greedy strategy is used to generate lane-change behavior, defined as $\pi(s) = \text{argmax}Q(s_t, a)$.

### C. Results

Experimental results are presented in this part, including reward, loss, and average Q value of different GRL algorithms. The rule-based method is selected as the baseline, and LC2013 lane-changing model is applied to AVs instead of the GRL algorithm.

*1) Reward:* During the training process, the reward for each training episode is shown in Fig.5. The training data

TABLE III

PARAMETERS OF GRL ALGORITHMS

| Parameters | Symbols | Value |
|---|---|---|
| Number of training episode | $N_{\text{training}}$ | 150 |
| Step size of the random exploration phase | $S_{\text{random}}$ | 20000 |
| Batch size | $M_{\text{batch}}$ | 32 |
| Replay buffer capacity | $M_{\text{replay}}$ | $10^6$ |
| Discount factor | $\gamma$ | 0.9 |
| Optimizer | - | Adam |
| Learning rate | $\eta$ | $10^{-4}$ |
| Online network update frequency | $N_{\text{online}}$ | 10 |
| Target network update frequency | $N_{\text{target}}$ | 1000 |
| Target network update method | - | soft |
| Soft target network update rate | $\eta_{soft}$ | 0.01 |

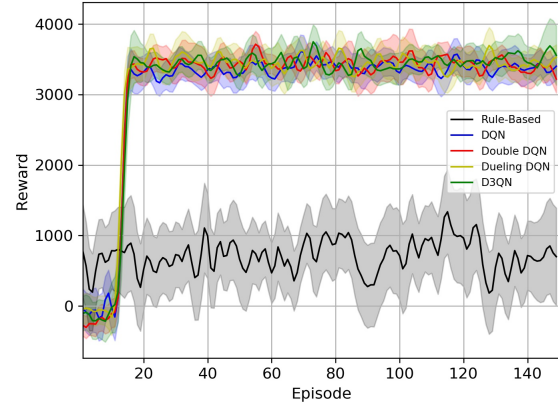from the end of the random exploration phase is summarized in Table IV.



Fig. 5. The reward curve of four GRL algorithms. The shaded areas show the standard deviation for 3 random seeds.

TABLE IV

TRAINING DATA OF FOUR GRL ALGORITHMS

| Algorithm | Average Training Reward | Standard Deviation |
|---|---|---|
| Rule-Based | 720.68 | 549.17 |
| DQN | 3375.50 | 208.40 |
| Double DQN | 3433.23 | 214.29 |
| Dueling DQN | 3453.67 | 220.64 |
| D3QN | 3460.90 | 222.88 |

*2) Loss:* During the training process, the loss is calculated for each training episode, which is shown in Fig.6.

*3) Average Q Value:* During the training process, the average of Q values in the Q table generated by the GRL network is calculated for each training episode to compare the estimation effects on the Q value of different algorithms. When the difference in rewards is small, a large average Q value implies an obvious overestimation effect of the algorithm. The average Q value for each training episode is shown in Fig.7.
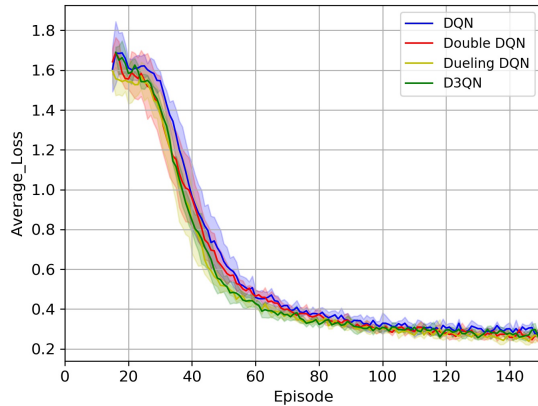
Fig. 6. The loss curve of four GRL algorithms. The shaded areas show the standard deviation for 3 random seeds.
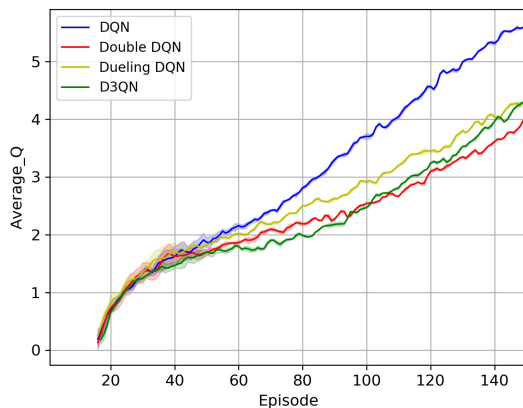


Fig. 7. The average Q value curve of four GRL algorithms. The shaded areas show the standard deviation for 3 random seeds.

## D. Analysis

In this part, experimental results are presented and analyzed to compare the performance of different GRL algorithms applied in our proposed framework.

As shown in Fig.5. The rewards of the four GRL algorithms are much higher than the rule-based method. It proves that the implementation of GNN can well represent the interaction between vehicles, and the combination of GNN and DRL can improve the performance of the generation of lane-change behaviors. In the random exploration phase, due to the random behavior generation of AVs, the reward is low with severe fluctuation; after the random exploration phase, the reward rises rapidly, and fluctuation becomes steady. However, the reward curve of the four algorithms didn't show significant differences. It can be concluded from Table. IV that the average reward of Double DQN is higher than that of DQN, and the average reward of Dueling DQN and D3QN is much higher than that of DQN. This is because the establishment of Dueling Network can better optimize the behavior generation of AVs.

As shown in Fig.6. The loss of the four algorithms in the training process decreases with the increase of the training episode and converges to a stable value. DQN and Double DQN have no obvious difference in the loss convergence process, but it is significantly higher than Dueling DQN and D3QN, and the convergence speed is relatively slow. After the loss convergence process, the loss curves of the four algorithms have no noticeable difference. It can be seen that there is no significant difference between the average loss of DQN and Double DQN. This result can prove that optimizing the strategy with Dueling Network can effectively reduce the loss in the training process.

It can be seen from Fig.7 that as the training episode increases, the average Q value increases continually; however, the variation trend of the average Q value for the four algorithms is quite different. The curve of DQN is significantly higher than the other three algorithms, but the average reward is relatively low; this is due to the problem of overestimation caused by the max operator when calculating the target Q value. The curve of Double DQN and D3QN changes relatively smoothly, which shows that the implementation of Double operation can effectively improve the effect of Q value evaluation. The average Q value of Dueling DQN is lower than DQN but higher than Double DQN and D3QN, indicating that the establishment of Dueling Network can also benefit the evaluation of Q value.

We then test the four GRL algorithms under the same experimental parameter settings for ten episodes. During the test process, we restricted the lane-changing behavior of all the vehicles to avoid collisions as much as possible. The testing data is calculated, and the results are shown in Table V.

TABLE V
TESTING DATA OF FOUR GRL ALGORITHMS

| Algorithm | Average Testing Reward | Standard Deviation |
|---|---|---|
| DQN | 3246.10 | 297.52 |
| Double DQN | 3283.57 | 225.34 |
| Dueling DQN | 3295.97 | 312.05 |
| D3QN | 3346.74 | 225.73 |

Results show that D3QN has the highest reward with stable standard deviation, followed by Dueling DQN, Double DQN, and DQN. It can be concluded that the improved DQN methods have shown promising results in the constructed traffic scenario.

## V. CONCLUSION

This research proposes a modular framework with high generality that consists of three key modules: scenario construction, GRL algorithm, and simulation. The proposed framework can be used to verify the effects of different GRL methods in various traffic scenarios according to the actual demands. A 3-lane highway with two ramp exits scenario is constructed, and four different types of GRL algorithms are applied and evaluated in our framework. The

experimental results show that the combination of GNN and DRL can well solve the decision-making task in an interactive traffic scenario. The designed framework can also realize the verification and comparison of different GRL algorithms in the field of intelligent transportation systems. Future work will focus on verification under the different parameters of the traffic flow, designing more discriminative traffic scenarios to verify the effect of the algorithm better, and implementing more GRL algorithms into the proposed framework.

## REFERENCES

[1] S. Triest, A. Villaflor, and J. M. Dolan, "Learning highway ramp merging via reinforcement learning with temporally-extended actions," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1595–1600.

[2] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3441–3447.

[3] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.

[4] Q. Liu, X. Li, S. Yuan, and Z. Li, "Decision-making technology for autonomous vehicles: Learning-based methods, applications and future outlook," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 30–37.

[5] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[6] P. Hart and A. Knoll, "Graph neural networks and reinforcement learning for behavior generation in semantic environments," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1589–1594.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1399–1404.

[9] J. Bernhard, S. Pollok, and A. Knoll, "Addressing inherent uncertainty: Risk-sensitive behavior generation for automated driving using distributional reinforcement learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2148–2155.

[10] C.-J. Hoel, T. Tram, and J. Sjöberg, "Reinforcement learning with uncertainty estimation for tactical decision-making in intersections," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[11] U. Yavas, T. Kumbasar, and N. K. Ure, "A new approach for tactical decision making in lane changing: Sample efficient deep q learning with a safety feedback reward," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1156–1161.

[12] Z. Li, J. Gong, C. Lu, and J. Xi, "Importance weighted gaussian process regression for transferable driver behaviour learning in the lane change scenario," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 497–12 509, 2020.

[13] Z. Li, J. Gong, C. Lu, and J. Li, "Personalized driver braking behavior modeling in the car-following scenario: An importance-weight-based transfer learning approach," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 10, pp. 10 704–10 714, 2022.

[14] F.-X. Devailly, D. Larocque, and L. Charlin, "Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[15] Q. Wang, "Varl: a variational autoencoder-based reinforcement learning framework for vehicle routing problems," *Applied Intelligence*, vol. 52, no. 8, pp. 8910–8923, 2022.

[16] Z. Li, J. Gong, C. Lu, and Y. Yi, "Interactive behavior prediction for heterogeneous traffic participants in the urban road: A graph-neural-network-based multitask learning framework," *IEEE/ASME Transactions on Mechatronics*, vol. PP, no. 99, p. 1, 2021.

[17] Z. Li, C. Lu, Y. Yi, and J. Gong, "A hierarchical framework for interactive behaviour prediction of heterogeneous traffic participants based on graph neural network," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[18] Z. Yu and M. Hu, "Deep reinforcement learning with graph representation for vehicle repositioning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[19] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *2020 International Conference on Learning Representations (ICLR)*, 2020.

[20] J. Dong, S. Chen, P. Y. J. Ha, Y. Li, and S. Labi, "A drl-based multiagent cooperative control framework for cav networks: a graphic convolution q network," *arXiv preprint arXiv:2010.05437*, 2020.

[21] X. Gao, X. Li, Q. Liu, Z. Li, F. Yang, and T. Luan, "Multi-agent decision-making modes in uncertain interactive traffic scenarios via graph convolution-based deep reinforcement learning," *Sensors*, vol. 22, no. 12, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/12/4586

[22] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, November 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/127994/

[23] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[24] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[28] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[29] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[31] Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa, "Chainerrl: A deep reinforcement learning library," *Journal of Machine Learning Research*, vol. 22, no. 77, pp. 1–14, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-376.html

[32] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[33] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *arXiv preprint arXiv:1710.05465*, vol. 10, 2017.

[34] M. Treiber and A. Kesting, "Traffic flow dynamics," *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[35] J. Erdmann, "Sumo's lane-changing model," in *Modeling Mobility with Open Data*. Springer, 2015, pp. 105–123.