Safe and Rule-Aware Deep Reinforcement Learning for Autonomous Driving at Intersections

Chi Zhang[1], Kais Kacem[2], Gereon Hinz[3], Alois Knoll[4]

***Abstract*— Driving through complex urban environments is a challenging task for autonomous vehicles (AVs), as they must safely reach their mission goal, and react properly to traffic participants while obeying traffic rules. Deep reinforcement learning (DRL) is a promising method to generate driving policies for AVs because it can explore complex environments and learn suitable reactions. In this work, we present a DRL algorithm for AVs to handle intersection scenarios while considering traffic rules. Furthermore, we enhance the safety of our DRL algorithm's decisions by introducing a safety checker based on a responsibility-sensitive safety (RSS) model. Evaluations show that our DRL algorithm outperforms the baseline method by driving safely to reach the mission goal while obeying the traffic rules at an intersection.**

I. INTRODUCTION

Safely driving through unsignalized intersections in urban areas is challenging for autonomous vehicles (AVs). To provide safe driving policies, AVs must interact with surrounding traffic participants and handle uncertainties such as noisy sensor data and inaccurate motion predictions, all while obeying traffic rules. In recent years, numerous studies have focused on applying deep reinforcement learning (DRL) in the decision-making tasks of AVs since it has successfully demonstrated superhuman-level performance in the fields of robotics and video games [1]. DRL facilitates learning of optimal long-term policies solving complex driving tasks, such as the handling of intersections, roundabouts, or lane changes on highways. Examples are presented by the authors of [2]–[5], who argue that DRL agents behave less conservatively compared to rule-based methods because of their negotiation and interaction capabilities.

In this work, we focus on applying DRL to learn driving policies for unsignalized intersections. In some countries, such as Germany, drivers must follow the right-before-left rule at intersections without traffic signs or traffic lights. For example, the ego vehicle must give way to the vehicles approaching from the right since they have higher priority (see Fig. 1). The DRL algorithm needs to safely drive through the intersection without harming other road users, while following the traffic rules. To train the DRL agent, a simulation environment is required that can provide high-fidelity vehicle dynamics for the ego vehicle, while simulating other vehicles that can comply with the right-before-left rule. Our goal is to address the aforementioned challenges to advance the safe and rule-compliant DRL algorithm toward real-world applications.

[1]Chi Zhang, [2]Kais Kacem are with ZF Friedrichshafen AG, Friedrichshafen, Germany. (chi.zhang@zf.com, kais.kacem@zf.com)

[3]Gereon Hinz, [4]Alois Knoll are with Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University of Munich, Germany. (gereon.hinz@tum.de, knoll@mytum.de)

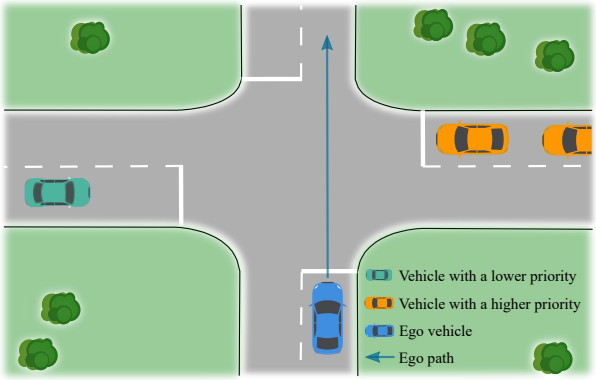

Fig. 1. The figure shows an intersection without traffic signs and traffic signals. The ego vehicle (blue car) intends to drive through the intersection. The yellow vehicles have priority over the ego vehicle, whereas the ego vehicle has the right of way over the green vehicle.

A. Related Work

1) Simulator: Several simulators have been designed specifically to develop autonomous driving systems and train DRL agents, as reviewed in [1]. Among them, "Simulation of Urban MObility" (SUMO) [6] and "Car Learning to Act" (CARLA) [7] are representative examples that have active developer teams and communities. By utilizing the traffic-management and intersection model of SUMO, users are able to apply traffic rules when defining the behavior of simulated road users. However, SUMO does not support the simulation of road user's physical model. In comparison, CARLA can simulate vehicles using a high-fidelity vehicle dynamics model.

2) Traffic rules for DRL: For autonomous driving tasks, traffic rules have been considered mostly to design reward functions for DRL agents. In highway scenarios, negative rewards are assigned if the DRL agent deviates from the center of its lane [8], [9]. Compared to highway scenarios, unsignalized intersections in urban environments are more challenging to the incorporation of traffic rules during DRL agent training. Some works assume that the ego vehicle always has the lowest priority [10], [11]. However, this simple assumption may cause issues such as overcautious driving behavior or deadlock situations. In [12], the DRL agent implicitly learns the right-of-way rule by interacting with other vehicles that comply with the traffic rules in the

simulation environment. However, no explicit information about the road priority is represented in the state space for the agent. The deployment of the DRL agent is constrained to the same priority setup as that in which it was trained.

Instead of making simple assumptions, another way to consider traffic rules is to explicitly incorporate information into the state space or reward function of the DRL agent. In [13], stop lines and yield lines are represented in the state space using a grid map with different colors. The positions of other vehicles and their priority levels are also embedded within the state space. [14] uses the priority information in both the state space and reward function. The DRL agent is penalized when it does not wait for the vehicles in the prioritized lanes. In addition, whether other vehicles have right of way over the DRL agent is modeled into the state space. However, the priority definitions and violation conditions are not discussed. Moreover, the impact of the traffic rule rewards on the DRL agent is not investigated.

3) Safety for DRL: Because the DRL agent is employed in a safety-critical system, it should learn driving policies that both comply with traffic rules and ensure safety. Several techniques have been proposed to improve the safety of DRL agent decisions in autonomous driving tasks. One promising direction is to utilize a safety model checker to check the action space and only provide safe actions during the DRL agent's training. In [15] and [16], prior knowledge and constraints are used for the DRL agent to maintain a safe distance and avoid lane changes that result in close distances to other vehicles. In [17], authors apply set-based predictions in a safety layer to remove the unsafe action candidates. [18] introduces a safety DRL method that utilizes system dynamics and recursive feasibility techniques to construct a supervision module that guarantees safety during learning. In [19], the authors apply regret theory to predict human drivers' lane-change decisions and use these predictions as safety constraints in the DRL training process. Instead of influencing the training process, another type of approach combines traditional methods with DRL. In [20], the vehicle is first trained with DRL to learn the driving policy in a static environment without other vehicles. Afterwards, an artificial potential field is employed to avoid collision with other vehicles. [21] proposes a hybrid method that combines the approximate partially observable Markov decision process with DRL to enhance safety.

B. Contributions

Driving safely and obeying traffic rules are fundamental requirements for applying the DRL algorithm in the real world. In this work, we propose a safe and rule-aware DRL approach for high-level decision making in an intersection scenario for autonomous driving. First, we establish a CARLA and SUMO co-simulation environment that can provide high-fidelity vehicle dynamics for the ego vehicle while simulating other vehicles that can comply with traffic rules, such as the right-before-left rule at intersections. Furthermore, we introduce a traffic rule monitor that checks the priority of the DRL-based ego vehicle according to the right-before-left rule. In addition to this, we apply a safety checker based on the responsibility-sensitive safety (RSS) model [22]. The safety checker verifies the status of the ego vehicle and provides a fallback safe action for the DRL agent in unsafe situations during the training and inference phases.

Our contributions are summarized below.

- We introduce a traffic rule monitor for detecting the compliance of the DRL agent with traffic rules.
- We apply an RSS-based safety checker to guarantee safety during the training and inference phases.
- We evaluate different approaches for combining the traffic rule monitor and the safety checker to achieve safe and rule-compliant intersection driving with the DRL approach.

The rest of this paper is structured as follows: The preliminary of the DRL is provided in Sec. II. Sec. III introduces the proposed DRL framework, including the CARLA and SUMO co-simulation, traffic rule monitor, and RSS-based safety checker. The simulation environment and setup are explained in Sec. IV. The evaluation results are provided in Sec. V. Finally, Sec. VI draws conclusions and discusses future work.

II. PRELIMINARY

Decision making with reinforcement learning can be modeled as a Markov decision process defined by the tuple (S, A, T, R, γ), where S is the state space, A is the action space, T is a state transition model that describes the transition probability from one state to another, R is a reward model, and $\gamma \in [0, 1)$ is a discount factor for calculating the cumulative expected reward. Following the transition model T, the environment transitions to a new state $s' \in S$ with a probability $T(s, a, s') = \Pr(s \mid s', a)$, and a reward $r = R(s, a)$ is assigned to the agent. The goal of the agent is to find the policy $\pi : S \to A$ that maximizes accumulated expected rewards by determining which action $a \in A$ should be taken for a given state $s \in S$.

The model-free reinforcement learning algorithm Q-learning [23], represents the policy π by a state-action value function $Q(s, a)$ where the optimal Q-function $Q^*(s, a)$ satisfies the Bellman equation:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a} Q^*(s',a). \quad (1)$$

In DRL, the state-action value function is represented by a neural network with the weights θ [24]. An approximate solution for (1) can be obtained by minimizing the following loss function:

$$L(\theta) = \mathbb{E}_{s'} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta) \right)^2 \right]. \quad (2)$$

III. SAFE AND TRAFFIC RULE-AWARE DEEP REINFORCEMENT LEARNING

In this section, we first briefly describe our safe and traffic rule-aware DRL framework. Then, we introduce the applied simulation environment. Following this, we explain our traffic rule monitor that checks for compliance with the

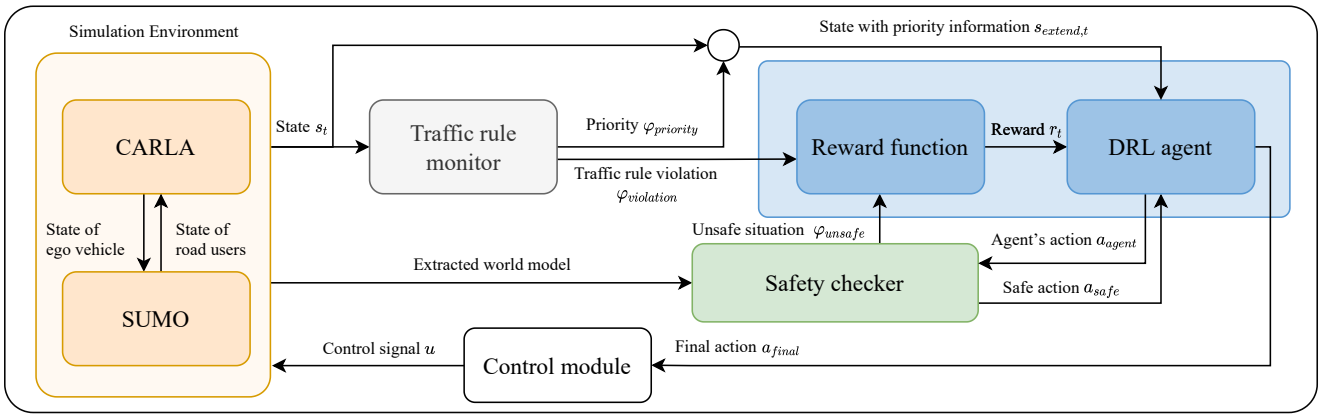

Fig. 2. Safe and rule-aware framework for DRL decision-making module of the autonomous vehicle.

right-of-way rule and discuss its integration into the state space and reward model of the DRL agent. Subsequently, we discuss the safety checker's design based on the RSS model. Finally, we explain the model of the DRL agent, including its state, action, and reward model.

A. Framework

In this study, we propose a DRL framework to make high-level decisions for AVs to handle intersections in urban environments (see Fig. 2). The task of the DRL agent is to safely drive the ego vehicle across intersections in compliance with the right-before-left rule. To train the DRL agent, we create an intersection scenario using both CARLA and SUMO simulators. At each time step, the state of the ego vehicle is updated by CARLA, while the state of other road users is updated by SUMO. The environment is synchronized between both simulators and provides the current state to the traffic rule monitor and safety checker.

The DRL agent should understand its priority over other vehicles and obey the right-before-left rule at intersections. Therefore, we introduce a traffic rule monitor to provide the priority information $\varphi_{priority}$ containing the priority relationship between the ego vehicle and other vehicles. The traffic rule monitor further checks whether the ego vehicle violates this rule. The priority information $\varphi_{priority}$ and the result of monitoring the traffic rule violation $\varphi_{violation}$ are further incorporated into the state space $s_{extend,t}$ and the reward function for the training of the DRL agent.

The safety checker verifies whether the current state is safe according to the RSS model. Given the agent's action a_{agent}, the safety checker returns a tuple $(\varphi_{unsafe}, a_{safe})$, where φ_{unsafe} is a Boolean value to indicate whether the current state is unsafe and a_{safe} is a safe fallback action. If the DRL agent doesn't choose a brake action in an unsafe situation, the safety checker interferes with the DRL agent's decision and provides a safe fallback action a_{safe} as the final action a_{final}. Finally, the control module converts the final action a_{final} to the control signal u for controlling the throttle or brake actuators in the AV. The reward function with traffic rule violation and unsafe information is used to guide the training of the DRL agent.

Fig. 3. A scenario in the CARLA simulator (left) and SUMO simulator (right). The states of the ego vehicle (blue car) and other road users are synchronized between CARLA and SUMO.

B. CARLA and SUMO Co-simulation

To train the DRL agent, we require a realistic environment for testability and integration into real-world applications. We apply SUMO to control road users so that they can comply with the right-of-way rule at the intersections. However, SUMO neither supports simulation of the vehicle's physical model nor includes sensor models. Therefore, we apply CARLA to simulate the environment and the ego vehicle with its physical behavior model. As shown in Fig. 3, both CARLA and SUMO can interact with each other by synchronizing the states of the ego vehicle and the other road users.

C. Traffic Rule Monitor

The traffic rule monitor checks if other vehicles have priority at an intersection according to the right-of-way rule, and evaluates if the ego vehicle violates this rule. In the following, we explain this evaluation in more detail.

In order to determine if the ego vehicle violates the right-of-way rule, we first need to find the vehicles driving on higher prioritized lanes with respect to the ego's current lane. We only consider intersections where the priority-relation between lanes does not change over time, i.e., intersections where no traffic lights are present. Thus, we obtain the priority-relation between lanes from a high-definition map.

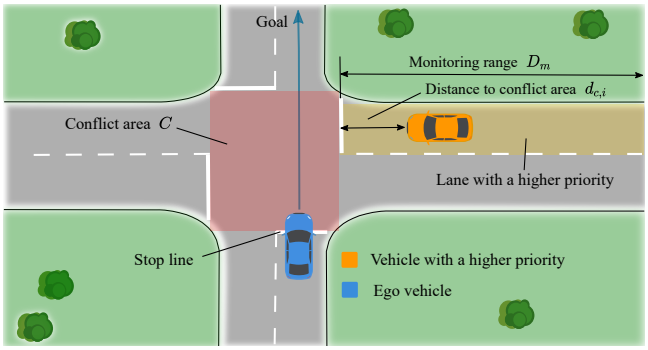

Fig. 4. Example of the ego vehicle violating the traffic rule in a right-before-left intersection.

For each intersection, we define a conflict area C to be the area enclosed by the stop lines (see. Fig. 4). In order to determine if the ego vehicle violates the priority rule, we only consider vehicles that satisfy the following priority condition:

$$\varphi_{priority,i} = \{d_{c,i} \leq D_m\} \vee \left\{ \frac{d_{c,i}}{v_i} \leq T_a \right\}, \qquad (3)$$

where $d_{c,i}$ is the Euclidean distance of vehicle i to the conflict area and v_i is its velocity. In this work, we define the arriving time threshold as $T_a = 3\ s$ and the monitoring range as $D_m = 30\ m$.

The traffic rule monitor returns $\varphi_{violation} = True$ if the ego vehicle has a geometrical overlap with the conflict area and if there is at least one vehicle i, for which the condition $\varphi_{priority,i} = True$ holds.

D. Safety Checker

The objective of the safety checker is to identify unsafe situations φ_{unsafe} and provide a proper reaction a_{safe} to ensure safety while guiding the DRL agent to learn how to behave properly in such situations. We integrate the open-source C++ library for RSS [25] within the safety checker in our DRL framework. The RSS model formalizes safe driving by defining a set of logical rules and mathematical formulas such as safe distance to other vehicles (see [22] for more details). Using these formalizations, the RSS model can identify dangerous situations, which we denote as $\varphi_{dangerous}$. The states of surrounding objects and the ego vehicle are extracted based on the simulation environment. As shown in Fig. 5, the extracted world model is the input for the RSS model to identify whether the situation is dangerous, i.e., $\varphi_{dangerous} = True$.

The action selection module is applied to decide whether to interfere with the agent's decision. In cases where the situation is safe according to the RSS model ($\varphi_{dangerous} = False$), or the DRL agent chooses a brake action a_{brake} when $\varphi_{dangerous} = True$, the action selection module forwards the DRL agent's action without any interference, such that $a_{safe} = a_{agent}$, and the checking result φ_{unsafe} is set to $False$. Otherwise, if the agent does not choose a braking action in a dangerous situation $\varphi_{dangerous} = True$, the action selection module in the safety checker sets $\varphi_{unsafe} = True$ and overwrites the agent's action with the braking action $a_{safe} = a_{brake}$. As a result, the safety checker provides the identified unsafe situation φ_{unsafe} to the reward function and the action a_{safe} to the DRL agent.

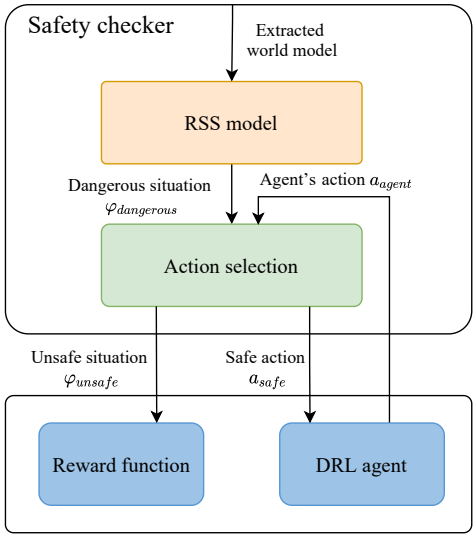

Fig. 5. The functional architecture of the safety checker based on the RSS model.

E. DRL Model

1) State Space: The state representation of the DRL agent is a feature list containing information about the ego vehicle and other road users. We include the five vehicles with the closest Euclidean distance to the conflict area of the intersection. The state representation with priority information at time step t is defined as $s_{extend,t} = [s_{ego}, s_{veh,1}, s_{veh,2}, s_{veh,3}, s_{veh,4}, s_{veh,5}]$. The state of the ego vehicle is defined as $s_{ego} = [v_{ego}, d_{goal}, d_{conflict}]$, which includes the ego vehicle's velocity v_{ego}, distance to the goal d_{goal}, and distance to the conflict area $d_{conflict}$.

The state of the other vehicles $s_{veh,i} = [v_i, d_{c,i}, \varphi_{priority,i}]$ includes the velocity of the vehicle v_i, distance to the conflict area $d_{c,i}$, and the priority information $\varphi_{priority,i}$. We add a negative sign to $d_{c,i}$ when a vehicle is driving away from the center of the intersection along its reference driving path. The priority information $\varphi_{priority,i}$ is set to $True$ if a vehicle i has a higher priority than the ego vehicle.

2) Action Space: The task of the DRL agent is to make high-level decisions in order to drive the AV through the intersection. Therefore, we apply high-level decisions represented by three discrete actions a_{drive}, $a_{cautious}$, and a_{brake}. The action a_{drive} indicates that the ego vehicle should drive with maximal velocity $v_{ego} = 5\ m/s$. The action $a_{cautious}$ commands the ego vehicle to drive with velocity $v_{ego} = 1\ m/s$. When a_{brake} is chosen, the ego vehicle should decelerate until $v_{ego} = 0\ m/s$. In our DRL framework, we apply a control module to convert the high-level decisions from the DRL agent to actuator control signals. These control signals include the percentage of throttle and brake applied in

the simulated ego vehicle in the co-simulation environment.

3) Reward Model: We define the total reward for the DRL agent in an episode as follows:

$$R = R_{collision} + R_{unsafe} + R_{efficiency} + R_{rule}. \quad (4)$$

The training of each episode ends if a collision is detected, the maximum allowed episode length is reached (time-out), or the ego vehicle reaches the goal.

$R_{collision}$ is the collision penalty. If the ego vehicle collides with another vehicle, a penalty is assigned based on the ego's velocity:

$$R_{collision} = -2v_{ego} - 5. \quad (5)$$

$R_{efficiency}$ is used to encourage the ego vehicle to finish each episode as quickly as possible. Therefore, a negative reward $R_{efficiency} = -0.1$ is given for each time step in a training episode.

R_{rule} is the reward for compliance with the right-of-way rule based on the checking result $\varphi_{violation}$ from the traffic rule monitor. For comparison purposes, we design two methods for rewarding the rule awareness of the DRL agent. The first method is a sparse rule violation reward $R_{rule,vi} = -5$, which is only assigned once in an episode when the ego vehicle violates the right-of-way rule given $\varphi_{violation} = True$. The second method is a dense rule-compliance reward $R_{rule,co}$, which encourages the ego vehicle to wait for other vehicles with a higher priority. For each time step when $\varphi_{violation} = False$, i.e., the ego vehicle waits behind the stop line to give way to vehicles with higher priority, it receives the positive reward $R_{rule,co} = 0.1$. In this case, the efficiency reward is reset to $R_{efficiency} = 0$, i.e., no penalty regarding efficiency is assigned when the ego vehicle complies with the traffic rule.

R_{unsafe} represents a penalty assigned at each time step $R_{unsafe} = -0.1$ if the ego vehicle is in an unsafe situation identified by the safety checker given $\varphi_{unsafe} = True$.

IV. IMPLEMENTATION

A. Simulation Environment

We implement the CARLA and SUMO co-simulation environment and the training pipeline on a computer with an Intel Xeon E5-2640 v4 processor running at 2.40 GHz and a Nvidia GeForce RTX 2080 Ti graphics processing unit. To investigate the effect of our DRL approach, we opted to use a double deep-Q network (DQN) [26] and prioritized experience replay [27] as the basis of the DRL agent. The policy network of the DRL agent is implemented using the PyTorch library [28]. The network structure and parameters are listed in Tab. I.

B. Simulation Setup

We train our DRL agent on an unsignalized four-way intersection. The other vehicles in the scenario comply with the right-before-left rule. At the beginning of the training phase, a random number of vehicles between one and ten are assigned to drive on one of the intersection lanes with different driving tasks: turning left, turning right, or going straight. At the beginning of each training episode, the ego vehicle is placed at its start position.

TABLE I
HYPER-PARAMETERS OF THE DEEP-Q NETWORK.

Parameter	Value	Description
α	2e 4	Learning rate for Adam optimizer
B_u	5000	Buffer size of the replay buffer
B_a	64	Batched samples from replay buffer
$[l_1, l_2, l_3]$	$[64, 64, 32]$	Hidden layer size
ϵ_{decay}	0.998	Epsilon decay
ϵ_{final}	0.01	Epsilon final value
γ	0.99	Discount factor
τ	1e 3	Soft target network update rate
Δt	0.1 s	Time step

V. EVALUATION

We evaluate our DRL framework with two experiments. In the first experiment, we design three different agents to evaluate the effect of the traffic rule monitor on the DRL agent. The second experiment investigates the advantage of further incorporating the safety checker into the DRL training pipeline. To train the DRL agents, we establish 60 s as the maximum allowed episode length. The metrics listed below are used to compare the performance of the DRL agents.

- Success rate: The average number of successful episodes. An episode is a success if the DRL agent reaches the goal within the maximum allowed episode length without collision.
- Collision rate: The average number of collisions. A collision is counted if the DRL agent collides with another vehicle.
- Infraction rate: The average number of infractions. An infraction occurs when the DRL agent violates the right-of-way rule, as checked by the traffic rule monitor within an episode.

A. Evaluation of the Traffic Rule Monitor

1) Experimental design: The goal of this experiment is to show the effect of our traffic rule monitor on the DRL agent. Therefore, we set up three DRL agents: a baseline DQN, and two DRL agents using our traffic rule monitor: RuleViDQN and RuleCoDQN.

DQN represents a standard DQN agent that implicitly explores and learns the right-of-way rule by interacting with other vehicles. Therefore, the DQN is trained only with the collision and efficiency rewards. Because this is the main configuration applied in previous works (e.g., [11], [12]), we consider the DQN as the baseline approach.

RuleViDQN denotes a violation-aware DQN combined with our traffic rule monitor. Compared to the standard DQN, the RuleViDQN has an additional reward $R_{rule,vi}$. The reward $R_{rule,vi}$ is a sparse reward that is only provided once in an episode when the DRL agent violates the right-of-way rule.

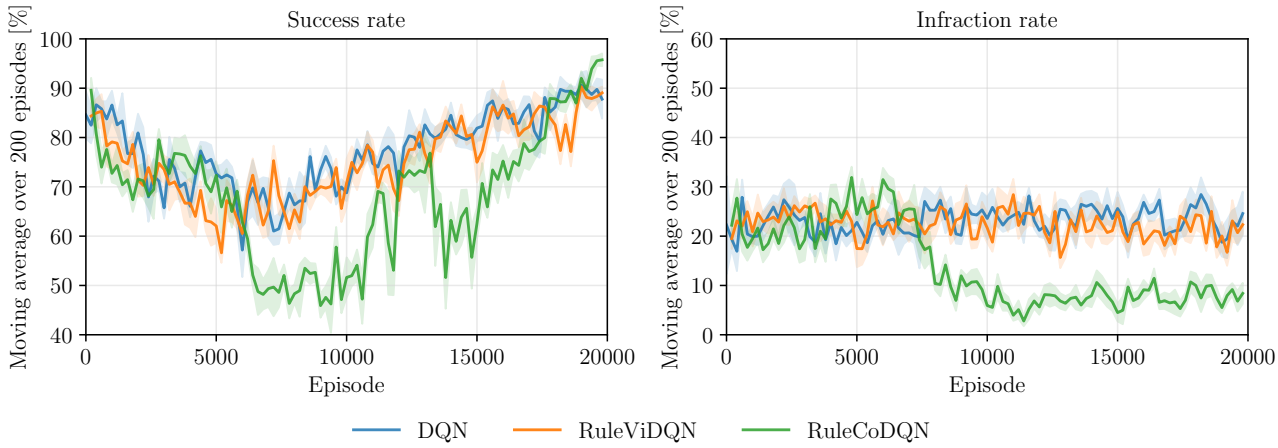

Fig. 6. Success and infraction rates of the three agents combined with the traffic rule monitor during the training. Using a dense reward to encourage agent adherence to the right-of-way rule, RuleCoDQN takes more episodes to learn but then significantly outperforms both DQN and RuleViDQN.

As suggested in [2], the sparse reward can be improved upon using dense reward by explicitly providing feedback for each time step. Therefore, we set up another DRL agent using our traffic rule monitor: **RuleCoDQN**, a rule-compliant DQN. In contrast to RuleViDQN, the RuleCoDQN has a dense reward $R_{rule,co}$. RuleCoDQN receives this positive reward $R_{rule,co}$ at each time step when the DRL agent adheres to the right-of-way rule.

2) Training performance: We calculate the moving average of success and infraction rates with a fixed subset size of 200 episodes and present each 200th average point in Fig. 6. At the start of the training, we can observe that the agents do not comply with the traffic rule but still often reach the goal. As the training progresses, the agents reduce the number of collisions by learning conservative driving policies. However, this also results in an initial decrease in the success rate since the maximum allowed episode length is reached frequently.

After 20,000 episodes of training, the standard DQN reaches the goal with around 90% success and 30% infractions. The high infraction rate reflects that the DRL agent cannot implicitly learn the traffic rule solely by exploring the world with other vehicles.

RuleViDQN has a similar moving average of success and infraction rates to the standard DQN, which indicates that the sparse reward does not guide the RuleViDQN to learn to obey the traffic rule. RuleViDQN fails to learn the rule because of the low number of episodes containing rule violation rewards.

The positive effect of encouraging the rule compliance on the RuleCoDQN can be seen from the green curves of the success and infraction rates in Fig. 6. After 6,000 episodes, the RuleCoDQN starts to comprehend the traffic rule and learns to obey it to obtain higher rewards. The success rate also drops after 6,000 episodes because the RuleCoDQN starts to explore the benefit of obeying the traffic rule guided by the positive reward $R_{rule,co}$. However, it waits too long to explore this benefit, which causes frequent time-outs and reduces the success rate. The increase in the success rate after 14,000 episodes reveals that the RuleCoDQN complies with the rule while reaching the goal.

TABLE II

EVALUATION RESULTS OF RULE-AWARE DRL AGENTS.

DRL Agent	Success Rate	Collision Rate	Infraction Rate
DQN	88.6%	10.9%	33.3%
RuleViDQN	93.3%	6.7%	26.5%
RuleCoDQN	**97.1%**	**2.9%**	**9.6%**

3) Running performance: Tab. II shows the success rate, collision rate, and infraction rate for the three agents during an evaluation with 2,000 episodes after training. The RuleCoDQN outperforms both the standard DQN and RuleViDQN, with the highest success rate of 97.1%, lowest collision rate of 2.9%, and lowest infraction rate of 9.6%. The evaluation results are similar to the training results, indicating that no overfitting occurred during training.

B. Evaluation of the Safety Checker

1) Experimental design: The safety checker provides two results: unsafe situation φ_{unsafe} and safe action a_{safe}. We design the following three variants of the DRL agent to compare their training performance. All three DRL variants are based on the RuleCoDQN, because it outperformed DQN and RuleViDQN in the previous experiment.

SafeReDQN uses the safety reward R_{unsafe} provided by the safety checker. It receives a penalty for each step when the ego is in an unsafe situation. **SafeAcDQN** uses the safe action provided by the safety checker instead of the original action, which may lead to an unsafe situation. **SafeReAcDQN** utilizes both the safety reward R_{unsafe} and the safe action during training. The goal is to guide the agent to identify unsafe situations and know how to react to them.

2) Training performance: Fig. 7 shows the training results in the same way as previously. The success and infraction rates of SafeReDQN do not improve during training, which indicates that the combination of the safety reward R_{unsafe} and the rule-compliance reward $R_{rule,co}$ cannot guide the

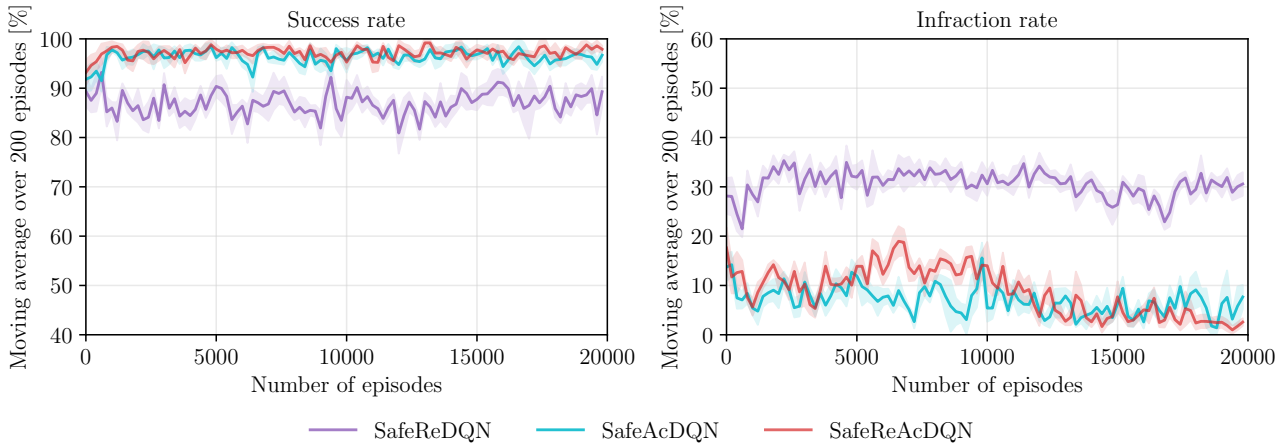

Fig. 7. Success and infraction rates of the three agents combined with the traffic rule monitor and safety checker during the training. SafeReAcDQN outperforms the other two agents with the best success rate and the lowest infraction rate at the end of training.

SafeReDQN properly. A conflict between these reward combinations occurs in the situation shown in Fig. 4, where the ego vehicle has crossed the stop line. In this situation, SafeReDQN would drive the ego vehicle further to avoid accumulating the penalty $R_{unsafe} = -0.1$ at each time step, instead of waiting for the other vehicle, which is more likely to cause a collision. Without other methods of guidance, SafeReDQN cannot learn how to react correctly in such a situation, resulting in more collisions and rule infractions.

Compared to SafeReDQN, both SafeAcDQN and SafeReAcDQN are protected with safe action a_{safe}, which prevents it from driving into an unsafe situation. The learning curves of SafeAcDQN and SafeReAcDQN converge faster than those of the other DRL agents. Moreover, SafeReAcDQN achieves the best success rate and the lowest infraction rate with smaller standard deviations compared to the other agents at the end of training (see Fig. 7); thus, the combination of the safety reward R_{unsafe} and the safe action a_{safe} provides the best guidance for training the DRL agent.

3) Running performance: We evaluate all trained agents over 2,000 episodes and summarize the results in Tab. III. Both SafeAcDQN and SafeReAcDQN benefit from the safe action and do not cause any collisions in the evaluation. Meanwhile, they drive more conservatively than the agents without safe action, with time-out rates of 3.6% and 1.4%, respectively. In summary, SafeReAcDQN performs best at reaching the mission goal successfully within the maximum allowed episode length, with the highest success rate of 98.6% and the lowest infraction rate of 2.6%. This result shows that SafeReAcDQN benefits the most from incorporating both a safety reward R_{unsafe} and a safe action a_{safe} from the safety checker. We provide a supplementary video[1] of the evaluated scenarios.

TABLE III

EVALUATION RESULTS OF SAFE AND RULE-AWARE DRL AGENTS.

DRL Agent	Success Rate	Collision Rate	Infraction Rate
SafeReDQN	88.2%	11.7%	28.2%
SafeAcDQN	96.4%	**0%**	4.5%
SafeReAcDQN	**98.6%**	**0%**	**2.6%**

Finally, to provide a more intuitive comparison of the effect of traffic rule monitor and safety checker, we show the training performance of the baseline DQN and the best agents from both evaluations, RuleCoDQN and SafeReAcDQN, in Fig. 8.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we present a safe and rule-aware DRL framework for autonomous driving at urban intersections. A co-simulation training environment combining CARLA and SUMO is applied to provide a realistic dynamic simulation of the ego vehicle and traffic participants complying with traffic rules. We investigate different ways to incorporate rule awareness into the DRL agent using our traffic rule monitor. Our experiments show that the DRL agent achieves better rule-compliance results using a dense reward than a sparse reward. Finally, we improve upon the rule-compliant agent by applying an RSS-based safety checker to ensure the safety of the DRL agent's driving policy. The evaluations show that the DRL agent with a safety reward and safe action guidance achieves the best performance with no collisions.

Future work includes the application of our approach to more scenarios, like different intersections, roundabouts, and highway merges. We also plan to include more traffic rules in the traffic rule monitor. Furthermore, we want to investigate the applicability of our approach in a multi-agent planning setup.

REFERENCES

[1] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *Journal of IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2022.

[2] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1205–1212.

[1] Video: `https://github.com/GitChiZhang/SafeReAcDQN`

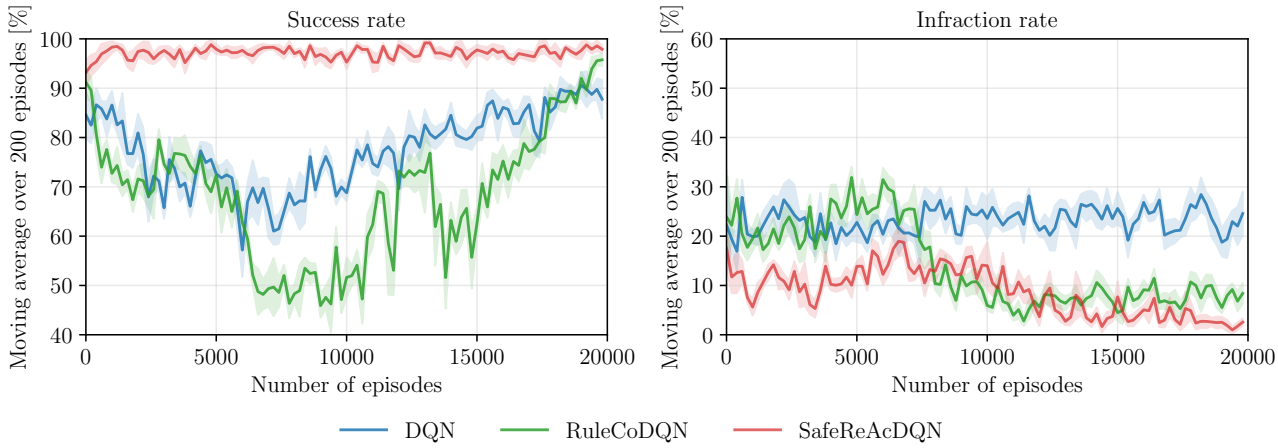

Fig. 8. Comparison of the training performance of DQN, RuleCoDQN, and SafeReAcDQN from evaluations A and B. By combining the traffic rule monitor and the safety checker, SafeReAcDQN achieves the best success rate and the lowest infraction rate compared to the baseline DQN and RuleCoDQN, which have only a traffic rule monitor.

[3] Z. Qiao, J. Schneider, and J. M. Dolan, "Behavior planning at urban intersections through hierarchical reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2667–2673.

[4] A. P. Capasso, G. Bacchiani, and D. Molinari, "Intelligent roundabout insertion using deep reinforcement learning," 2020, arXiv preprint arXiv:2001.00786.

[5] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement learning with iterative reasoning for merging in dense traffic," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[6] 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 *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proc. of conference on robot learning*, 2017, pp. 1–16.

[8] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker, "Dynamic input for deep reinforcement learning in autonomous driving," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 7566–7573.

[9] L. Chen, Y. Chen, X. Yao, Y. Shan, and L. Chen, "An adaptive path tracking controller based on reinforcement learning with urban driving application," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2411–2416.

[10] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2018, pp. 584–599.

[11] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[12] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," 2019, arXiv preprint arXiv:1911.12250.

[13] A. P. Capasso, P. Maramotti, A. Dell'Eva, and A. Broggi, "End-to-end intersection handling using multi-agent deep reinforcement learning," 2021, arXiv preprint arXiv:2104.13617.

[14] C. Li and K. Czarnecki, "Urban driving with multi-objective deep reinforcement learning," 2018, arXiv preprint arXiv:1811.08586.

[15] D. Liu, M. Brännstrom, A. Backhouse, and L. Svensson, "Learning faster to perform autonomous lane changes by constructing maneuvers from shielded semantic actions," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2019, pp. 1838–1844.

[16] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2156–2162.

[17] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–7.

[18] Z. Li, U. Kalabić, and T. Chu, "Safe reinforcement learning: Learning with supervision using a constraint-admissible set," in *Proc. of American Control Conference (ACC)*, 2018, pp. 6390–6395.

[19] D. Chen, L. Jiang, Y. Wang, and Z. Li, "Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model," in *Proc. of American Control Conference (ACC)*, 2020, pp. 4355–4361.

[20] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," 2016, arXiv preprint arXiv:1612.00147.

[21] F. Pusse and M. Klusch, "Hybrid online pomdp planning and deep reinforcement learning for safer self-driving cars," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1013–1020.

[22] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2017, arXiv preprint arXiv:1708.06374.

[23] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[24] 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.

[25] B. Gassmann, F. Oboril, C. Buerkle, S. Liu, S. Yan, M. S. Elli, I. Alvarez, N. Aerrabotu, S. Jaber, P. van Beek, *et al.*, "Towards standardization of av safety: C++ library for responsibility sensitive safety," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2265–2271.

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

[27] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, arXiv preprint arXiv:1511.05952.

[28] 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," *Proc. of Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.