

Deep Multi-agent Reinforcement Learning for Highway On-Ramp Merging in Mixed Traffic

Dong Chen¹, Zhaojian Li^{*1}, Yongqiang Wang², Longsheng Jiang³, Yue Wang³

Abstract—On-ramp merging is a challenging task for autonomous vehicles (AVs), especially in mixed traffic where AVs coexist with human-driven vehicles (HDVs). In this paper, we formulate the mixed-traffic highway on-ramp merging problem as a multi-agent reinforcement learning (MARL) problem, where the AVs (on both merge lane and through lane) collaboratively learn a policy to adapt to HDVs to maximize the traffic throughput. We develop an efficient and scalable MARL framework that can be used in dynamic traffic where the communication topology could be time-varying. Parameter sharing and local rewards are exploited to foster inter-agent cooperation while achieving great scalability. An action masking scheme is employed to improve learning efficiency by filtering out invalid/unsafe actions at each step. In addition, a novel priority-based safety supervisor is developed to significantly reduce collision rate and greatly expedite the training process. A gym-like simulation environment is developed and open-sourced with three different levels of traffic densities. We exploit curriculum learning to efficiently learn harder tasks from trained models under simpler settings. Comprehensive experimental results show the proposed MARL framework consistently outperforms several state-of-the-art benchmarks.

Index Terms—Multi-agent deep reinforcement learning, connected autonomous vehicles, safety enhancement, on-ramp merging.

I. INTRODUCTION

Autonomous vehicle (AV) technologies, such as Tesla Autopilot [1] and Baidu Apollo [2], have already been deployed in (semi-)autonomous vehicles on real-world roads. Despite the great advances over the past decade that have made this possible, the number of traffic accidents involving AVs are increasing in recent years [3], [4]. The accidents are often caused by the inability of AVs to timely react to the dynamic driving environment, especially in a mixed traffic with both AVs and human-driven vehicles (HDVs); the AVs need not only to react to road objects but also to attend to the behaviors of HDVs. Among the many challenging driving scenarios, highway on-ramp merging is one of the most difficult tasks for AVs [5], [6], which is the topic of this paper.

The considered on-ramp merging scenario is illustrated in Fig. 1, where we consider a general setup that AVs and

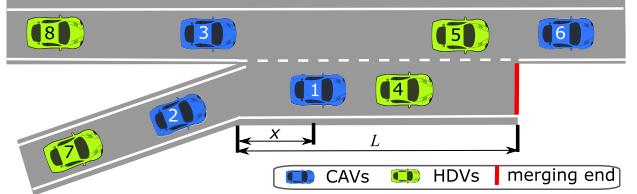


Figure 1: Illustration of the considered on-ramp merging traffic scenario. CAVs (blue) and HDVs (green) coexist on both ramp and through lanes.

HDVs coexist on both merge lane and through lane. On-ramp vehicles need to efficiently merge onto the through lane without collision. In an ideal cooperative setting, the vehicles on the through lane should proactively decelerate or accelerate to make adequate space for on-ramp vehicles to safely merge whereas the on-ramp vehicles also adjust speed and promptly cut in when it is safe, to avoid deadlock situations [7]. It is clear that coordination between the vehicles is a crucial enabler for safe and efficient merging maneuvers. While this is relatively easy to achieve in a full-AV scenario, AV coordination in the presence of HDVs is a significantly more challenging task.

Rule-based and optimization-based methods have been proposed to tackle the automated merging problem [8]–[11]. In particular, rule-based approaches employ heuristics and hard-coded rules to guide the AVs [9], [10]. While this is feasible for simple traffic scenarios, these methods quickly become impractical in more complex merging scenarios [12]. In an optimal control setting, vehicle interactions are modeled as a dynamic system with actions from controlled vehicles as inputs. For example, a model predictive control (MPC) approach is developed to control an AV to merge in a parallel-type ramp [12]. While promising results are demonstrated, the MPC-based methods rely on accurate dynamic merging models (including human driving models) and are typically computationally-involved as online optimizations are needed at each time step [13].

On the other hand, data-driven methods such as reinforcement learning (RL) have received increased attention and been explored for AV highway merging [11], [14]. Specifically, a multi-objective reward function for safety and jerk minimization is designed for AV merging and the Deep Deterministic Policy Gradient (DDPG) algorithm [15] is exploited to solve the RL problem in [11]. In [14], RL and MPC are integrated to promote the learning efficiency, which achieves a good trade-off between passenger comfort, efficiency, crash rate, and

¹ Dong Chen and Zhaojian Li are with the Department of Mechanical Engineering, Michigan State University, Lansing, MI, 48824, USA. Email: {chendon9, lizhaoj1}@msu.edu.

² Yongqiang Wang is with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29630, USA. Email: yongqiw@clemson.edu.

³ Longsheng Jiang and Yue Wang are with the Department of Mechanical Engineering, Clemson University, Clemson, SC, 29634, USA. Email: {longshj, yue6}@g.clemson.edu.

* Zhaojian Li is the corresponding author.

robustness. However, those approaches are only designed for a single AV, treating all other vehicles as part of the environment.

In this paper, we treat a general setup (see Fig. 1) where multiple AVs learn to adapt to HDVs and cooperatively accomplish merging tasks to maximize traffic throughput safely. As a result, it is natural to extend the single-agent RL to a multi-agent reinforcement learning (MARL) framework where the AVs collaboratively learn control policies to achieve the aforementioned goal (see Section II.B for a review of state-of-the-art MARL algorithms). However, this is a challenging task due to dynamic connectivity topology, complex motion patterns involving AV coupled dynamics, and intricate decision makings. This complexity is even more pronounced when human drivers are involved.

While several MARL approaches have been developed for connected and autonomous vehicles (CAVs) in car-following and lane overtaking scenarios [16]–[22], to the best of our knowledge, no MARL algorithm has been developed for the considered highway on-ramp merging scenario. In this work, we develop a novel decentralized MARL framework to enable AVs to efficiently learn a safe and efficient policy in the highway on-ramp merging scenario where a general policy is learned for vehicles on both lanes. A priority-based safety supervisor is designed to enhance safety and improve learning efficiency, through sequential and multi-step predictions. Parameter sharing and local rewards are exploited to foster inter-agent cooperation while achieving great scalability. The main contributions and the technical advancements of this paper are summarized as follows.

- 1) We formulate the mixed-traffic on-ramp merging problem (with AVs and HDVs coexisting on both ramp and through lanes) as a decentralized MARL problem. The formulation can allow for a dynamic environment with a time-varying connectivity topology. A corresponding gym-like simulation platform with three different levels of traffic density is developed and open-sourced¹.
- 2) We develop a novel, efficient, and scalable MARL algorithm, featuring a parameter-sharing mechanism, effective reward function design, and action masking. Furthermore, a priority-based safety supervisor is developed, which significantly reduces collision rates in training and subsequently improves learning efficiency.
- 3) We employ curriculum learning to speed up the learning for harder tasks by building upon trained models from less complex traffic scenarios.
- 4) We conduct comprehensive experiments, and the results show that the proposed approach consistently outperforms several state-of-the-art algorithms in terms of driving safety and efficiency.

The remainder of the paper is organized as follows. Section II briefly introduces RL and MARL, and reviews state-of-the-art algorithms. The problem formulation and the proposed MARL framework are described in Section III whereas the priority-based safety supervisor is detailed in Section IV. Experiments, results, and discussions are presented in Section V. We conclude the paper and discuss future works in Section VI.

¹See https://github.com/Derekabc/MARL_CAVs

II. BACKGROUND

In this section, we review the preliminaries of RL and introduce several state-of-the-art MARL algorithms to put our proposed work in proper context.

A. Preliminaries of Reinforcement Learning (RL)

In a RL setting, at each time step t , the agent observes the state $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$, takes an action $a_t \in \mathcal{A} \subseteq \mathbb{R}^m$, and subsequently receives a reward signal $r_t \in \mathbb{R}$ and an updated state s_{t+1} at time $t + 1$ from the environment. The goal of the RL agent is to learn an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$, a mapping from state to action, that maximizes the accumulated reward $R_t = \sum_{k=0}^T \gamma^k r_{t+k}$, where r_{t+k} is the reward at time step $t + k$ and $\gamma \in (0, 1]$ is the discount factor that quantifies the relative importance one wants to place on future rewards.

The state-action value function (or Q-function) under policy π , denoted by $Q^\pi(s_t, a_t)$, is an estimate of the expected return (accumulated reward in an infinite horizon) if starting from state s_t , taking an immediate action a_t , and then following policy π afterwards. The optimal Q-function can be characterized by the following Bellman equation, $Q^*(s_t, a_t) = E[r(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})]$, where the next state s_{t+1} is sampled from the environment's transition rules $P(s_{t+1}|s_t, a_t)$. The state value function of a state s_t under policy π , $V^\pi(s_t)$, is defined as the expected return if starting from s_t and immediately following policy π , i.e., $V^\pi(s_t) = E_\pi[R_t|s_t = s]$. Often the agent's policy is parameterized by some parameters θ and the goal is to learn appropriate θ to achieve desired system behavior. In actor-critic (A2C) algorithms [23], two networks are employed: a critic network parameterized by ϕ to learn the value function $V_\phi^{\pi_\theta}(s_t)$ and an actor network $\pi_\theta(a_t|s_t)$ parameterized by θ . The policy network is updated by maximizing the following objective function:

$$J^{\pi_\theta} = E_{\pi_\theta} [\log \pi_\theta(a_t|s_t) A_t], \quad (1)$$

where $A_t = Q^{\pi_\theta}(s_t, a_t) - V_\phi(s_t)$ is the advantage function that characterizes the improvement on reward if taking action a_t over the average reward of all possible actions taken at state s_t [23]. The value function parameter ϕ is updated by minimizing the following loss function:

$$J^{V_\phi} = \min_\phi E_{\mathcal{D}} \left(R_t + \gamma V_{\phi'}(s_{t+1}) - V_\phi(s_t) \right)^2, \quad (2)$$

where \mathcal{D} denotes an experience replay buffer that collects previously encountered experiences and ϕ' denotes the parameters obtained from earlier iterations used in a target network [24].

B. Multi-agent Reinforcement Learning (MARL)

MARL has found great successes across a wide range of multi-agent systems, including traffic light control [25], games [26], resource management in wireless networks [27], and powergrid control [28], only to name a few. MARL algorithms can be categorized into two main classes: *cooperative* and *non-cooperative*. In this paper, we will focus on the *cooperative* setting where all agents are encouraged to cooperate to achieve a common goal, i.e., safely maneuver

with maximum throughput. We next introduce a few state-of-the-art cooperative MARL algorithms that we will use as benchmarks for comparison in Section V.

An independent MARL framework, called IQL, is proposed in [29], allowing each agent to learn independently and simultaneously while viewing other agents as part of the environment. While fully scalable, it suffers from non-stationarity and partial observability. An off-policy MARL algorithm is proposed in [30] where collaboration is achieved by estimating the state-action value function using a centralized critic network based on global observations and actions. In [25] a learnable communication protocol and a spatial distance factor are proposed to scale down the reward signals of neighboring agents during training. Experimental results show good scalability and improved cooperation among agents. However, these MARL approaches only consider a stationary environment with fixed communication topology and thus the algorithms need to be re-designed and/or re-trained whenever the communication typology changes.

Recently, parameter sharing is widely applied in MARL settings with homogeneous agents [18], [31], [32], which bootstraps single-agent RL methods and learns an identical policy for each agent, and thus enables the handling of changes in the number of participating agents. In [32], several state-of-the-art single RL algorithms (i.e., PPO [33] and ACKTR [34]) are extended to the MARL with parameter sharing denoted as MAPPO and MAACKTR. A parameter sharing A2C (MA2C) algorithm is proposed in [31] to solve the fleet management problem and experimental results are given to confirm the performance. These methods will be used as benchmarks for performance comparison in Section V.

Several recent works also address safety issues in MARL problems. For example, a centralized shielding approach is introduced in [35], where a centralized model is used to monitor the joint actions of all agents and restrict unsafe actions. To address the scalability problem of centralized supervision, a local shielding approach is developed for only a subset of agents. Experiments in two-player navigation games in the grid world show good performance on collision avoidance. In addition, [36] proposes a decentralized control barrier function which shields unsafe actions based on available local information. They demonstrate the performance of proposed approach using patrol tasks where two agents navigate in an environment with obstacles and walls. However, these methods consider an all-autonomous, cooperative agent environment, without considering moving objects like HDVs.

To fill the aforementioned gaps, in this paper, we develop a novel on-policy MARL algorithm for the considered on-ramp merging problem with great efficiency and safety, which features action masking, priority-based safety supervisor, parameter sharing, and local reward shaping. Performance comparison between the proposed algorithm and the above benchmarks are presented in Section V.

III. RAMP MERGING AS MARL

In this section, we first formulate the considered on-ramp merging problem as a partially observable Markov decision

process (POMDP) [37]. Then we present our actor-critic-based MARL algorithm, featuring a parameter-sharing mechanism, effective reward function design, and action masking, to solve the formulated POMDP, which is denoted as the baseline method in Section V.

A. Problem Formulation

In this paper, we model the on-ramp merging environment in a mixed traffic as a model-free multi-agent network [25], [28], $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each agent $i \in \mathcal{V}$ communicates with its neighbors $\mathcal{N}_i := \{j | \varepsilon_{ij} \in \mathcal{E}\}$ through the edge connections $\varepsilon_{ij}, i \neq j$. Let $S := \times_{i \in \mathcal{V}} S_i$ and $\mathcal{A} := \times_{i \in \mathcal{V}} \mathcal{A}_i$ denote the global state space and action space, respectively. The underlying dynamics of the system can be characterized by the state transition distribution $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. We consider a decentralized MARL framework where each agent i (AV i) only observes a part of the environment (i.e., surrounding vehicles). This is consistent with the reality that AVs can only sense or communicate with vehicles in the close vicinity, making the overall dynamical system a POMDP $\mathcal{M}_{\mathcal{G}}$, which can be described by the following tuple $(\{\mathcal{A}_i, \mathcal{S}_i, \mathcal{R}_i\}_{i \in \mathcal{V}}, \mathcal{T})$:

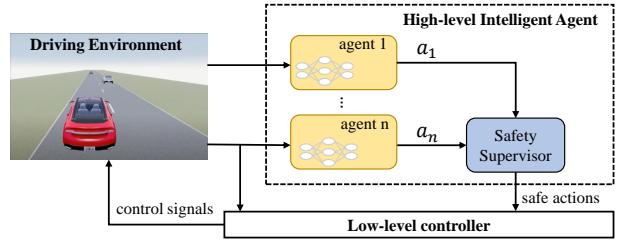


Figure 2: Schematics of system and simulation setup.

- **Action Space:** The action space \mathcal{A}_i of agent i is defined as the set of high-level control decisions, including *turn left*, *turn right*, *idle*, *speed up*, and *slow down*. With a selected high-level decision, lower-level controllers will then produce the corresponding steering and throttle control signals to maneuver the AVs. The system and simulation setup is illustrated in Fig. 2. The overall action space of the system is the joint actions from all AVs, i.e., $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_N$.
- **State Space:** The state of agent i , \mathcal{S}_i , is defined as a matrix of dimension $N_{\mathcal{N}_i} \times W$, where $N_{\mathcal{N}_i}$ is the number of observed vehicles and W is the number of features used to represent the state of a vehicle, including:
 - *ispresent*: a binary variable denoting whether a vehicle is observable in the vicinity of the ego vehicle.
 - x_l : the longitudinal position of the observed vehicle relative to the ego vehicle.
 - y : the lateral position of the observed vehicle relative to the ego vehicle.
 - v_x : the longitudinal speed of the observed vehicle relative to the ego vehicle.
 - v_y : the lateral speed of the observed vehicle relative to the ego vehicle.

We assume that only the “neighboring vehicles” can be observed by the ego vehicle. The “neighboring vehicles”

are defined as the vehicles on the same lane or adjacent lane(s) that are within a 150 m longitudinal distance from the ego vehicle. In the considered two-lane scenario (see Fig. 1), the neighboring vehicles include *same-lane front*, *same-lane behind*, *adjacent-lane front*, and *adjacent-lane behind*, and we thus have $N_{\mathcal{N}_i} = 5$ (4 neighbors plus the ego vehicle). In the example of Fig. 1, the neighboring vehicles of vehicle 1 are vehicles 2–5. The entire state of the system is then the Cartesian product of the individual states, i.e., $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \dots \times \mathcal{S}_N$.

- **Reward Function:** The reward function \mathcal{R}_i is crucial to train the RL agents so that it follows desired behaviors. As the objective is to train our agents to safely and efficiently pass the merging area, the reward for the i th agent at time step t is defined as follows:

$$r_{i,t} = w_c r_c + w_s r_s + w_h r_h + w_m r_m, \quad (3)$$

where w_c , w_s , w_h , and w_m are positive weighting scalars corresponding to collision evaluation r_c , stable-speed evaluation r_s , headway time evaluation r_h , and merging cost evaluation r_m , respectively. As safety is the most important criteria, we make w_c much bigger than other weights to prioritize safety. The four performance metrics are defined as follows:

- the collision evaluation r_c is set to -1 if collision happens, otherwise $r_c = 0$.
- the speed evaluation r_s is defined as

$$r_s = \min \left\{ \frac{v_t - v_{\min}}{v_{\max} - v_{\min}}, 1 \right\}, \quad (4)$$

where v_t , $v_{\min} = 20 \text{ m/s}$, and $v_{\max} = 30 \text{ m/s}$ are the current, minimum, and maximum speeds of the ego vehicle, respectively.

- the time headway evaluation is defined as:

$$r_h = \log \frac{d_{\text{headway}}}{t_h v_t}, \quad (5)$$

where d_{headway} is the distance headway and t_h is a predefined time headway threshold. As such, the ego vehicle will get penalized when the time headway is less than t_h and rewarded only when the time headway is greater than t_h . In this paper, we choose t_h as 1.2 s as suggested in [38].

- The merging cost r_m is designed to penalize the waiting time on the merge lane to avoid deadlocks [7]. Here we adopt $r_m = -\exp(-(x - L)^2/10L)$, where x is the distance the ego vehicle has navigated on the ramp and L is the length of the ramp (see Fig. 1). The merging cost function is plotted in Fig. 3, which shows that the penalty increases as the ego vehicle moves closer to the merging end.

- **Transition Probabilities:** the transition probability $\mathcal{T}(s'|s, a)$ characterizes the dynamics of the system. In the developed simulator, we exploit the intelligent driver model (IDM) [39] and MOBIL model [40] for longitudinal acceleration and lateral lane change decisions of HDVs, respectively. The high-level decisions of AVs are made by the MARL algorithm and will be tracked by

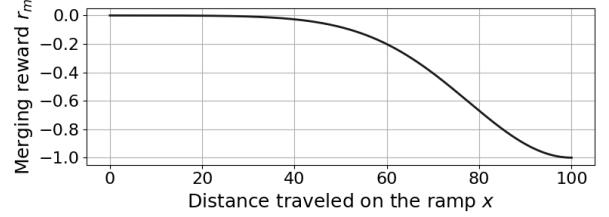


Figure 3: Illustration of the designed merging reward/penalty.

the lower-level controller (PID controller) (see Fig. 2). A kinematic bicycle model [41] is used to propagate vehicle trajectories. We do not assume any prior knowledge of the transition probability in the development of our MARL algorithm.

B. MA2C for CAVs

In the cooperative MARL setting, the objective is to maximize the global reward $R_{g,t} = \sum_{i=1}^N r_{i,t}$. Ideally, each agent will be assigned with the same average global reward $R_t = \frac{1}{N} R_{g,t}$ during training, i.e., $r_{1,t} = r_{2,t} = \dots = r_{N,t}$. However, this shared reward approach does not accurately indicate the contributions of each vehicle and can lead to several issues [42], [43]. First, aggregating the global reward can cause large latency and increase the communication overheads, which is problematic for systems with real-time constraints such as AVs. Second, a single global reward leads to the credit assignment problem [44], which can significantly impede the learning efficiency and limit the number of agents to a small size. Therefore, in this paper, we adopt a local reward assignment strategy, where each ego vehicle is only affected by its neighboring vehicles. Specifically, the reward for the i th agent at time t is defined as:

$$r_{i,t} = \frac{1}{|\nu_i|} \sum_{j \in \nu_i} r_{j,t}, \quad (6)$$

where $\nu_i = i \cup \mathcal{N}_i$ is a set containing the ego vehicle and its neighbors, and $|\cdot|$ denotes the cardinality of a set. This local reward design only includes rewards from agents that are most related to the success or failure of a task [31], [45]. This is appropriate for on-road vehicles as a vehicle only interacts with its surrounding vehicles and distant vehicles have limited impact on the ego vehicle.

The used network backbone is shown in Fig. 4, where the actor network and the critic network share the same low-level representations, and the policy loss and the value function error loss are thus combined into a single loss function [33]. The performance comparison between shared and separate actor-critic networks will be discussed in Section V-A. With the shared network parameters, the overall loss function takes the following form:

$$J(\theta_i) = J^{\pi_{\theta_i}} - \beta_1 J^{V_{\phi_i}} + \beta_2 H(\pi_{\theta_i}(s_t)), \quad (7)$$

where β_1 and β_2 are the weighting coefficients for the value function loss and the entropy regularization term, $H(\pi_{\theta_i}(s_t)) = E_{\pi_{\theta_i}}[-\log(\pi_{\theta_i}(s_t))]$, used to encourage the

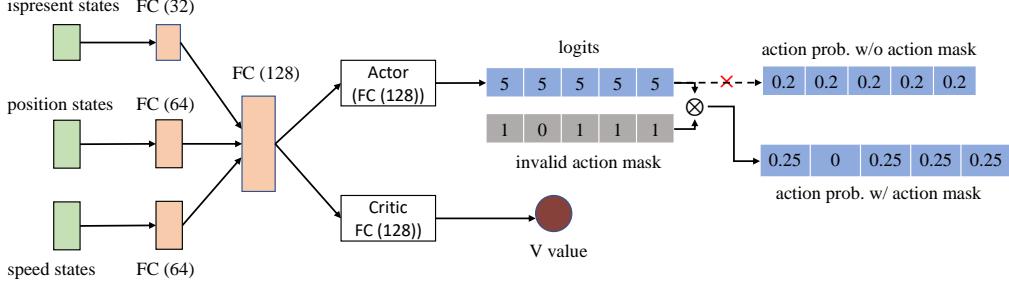


Figure 4: Architecture of the proposed network structure. The numbers in the parentheses denote the size of the layers.

agents to explore new states [33], [46], respectively. From Eqn. 1, it follows that the policy loss can be written as:

$$J^{\pi_{\theta_i}} = E_{\pi_{\theta_i}} [\log \pi_{\theta_i}(a_{i,t}|s_{i,t}) A_{i,t}^{\pi_{\theta_i}}], \quad (8)$$

where $A_{i,t}^{\pi_{\theta_i}} = r_{i,t} + \gamma V^{\pi_{\phi_i}}(s_{i,t+1}) - V^{\pi_{\phi_i}}(s_{i,t})$ is the advantage function and $V^{\pi_{\phi_i}}(s_{i,t})$ is the state value function. The loss for updating the state value V_{ϕ_i} is in the following form:

$$J^{V_{\phi_i}} = \min_{\phi_i} E_{\mathcal{D}_i} [r_{i,t} + \gamma V_{\phi_i}(s_{i,t+1}) - V_{\phi_i}(s_{i,t})]^2. \quad (9)$$

We use separate experience reply buffers for each agent but the same policy network is updated with the same network parameters among agents. This is suitable as we train a general policy for both on-ramp and through AVs [18], [31]. Minibatches of sampled trajectories are exploited to update the network parameters using Eqn. 7 to reduce the variance.

C. DNN Settings

The deep neural network design is illustrated in Fig. 4. Specifically, to improve scalability and robustness, we regroup the observation $s_{i,t}$ according to their physical units. For instance, the observation $s_{i,t}$ is divided into three groups: $s_{i,t}^1 \cup s_{i,t}^2 \cup s_{i,t}^3$ according to their units, representing *ispresent states*, *position states* and *speed states*, respectively. Each of the three sub-state vectors is encoded by one fully connected (FC) layer and the three encoded states are then concatenated into a single vector. The concatenated vector is fed into a 128-neuron FC layer, the result of which is consumed by both the actor network and the critic network. In a standard setting, the logits l_i from the actor network will go to a Softmax layer, producing the probability by $\pi_{\theta_i}(s_i) = \text{softmax}([l_1, l_2, l_3, l_4, l_5])$, which is used to sample the actions, i.e., $a_i \sim \pi_{\theta_i}(s_i)$.

However, this sampling procedure has several issues. First, invalid/unsafe actions are also assigned with non-zero probabilities; as a stochastic policy is used, these unsafe actions may be sampled during training, which can lead to undesirable system behaviors and even system breakdown. Second, sampling invalid/unsafe actions also impedes policy training as invalid policy updates [47] are executed for invalid actions, since the collected experiences associated with the invalid actions are not meaningful and misleading. To address these issues, we adopt the invalid action masking approach [31] which “masks out” invalid actions and only samples from valid actions. As shown in Fig. 4, with an invalid action mask obtained from

the environment (e.g., based on the traffic scenario) where “0” represents an invalid action and “1” denotes a valid action, the corresponding logits of invalid actions are replaced with large negative values, e.g., $-1e8$. As a result, the probability of the invalid actions after the Softmax layer is very close to 0, and sampling from invalid actions can thus be avoided, equivalently “renormalizing the probability distribution” [32]. In this work, we consider the following invalid actions:

- the ego vehicle attempts to make lane changes to a non-existing lane. For example, the ego vehicle tries to make a left turn when it is already on the leftmost lane.
- the ego vehicle attempts to speed up or slow down when its speed has already reached the predefined maximum or minimum speed.

IV. PRIORITY-BASED SAFETY ENHANCEMENT

While obvious invalid actions can be avoided using the rule-based action masking scheme described above, it cannot prevent inter-vehicle or vehicle-obstacle collisions. Therefore, a more comprehensive safety supervisor is needed to deal with collisions in complex, dynamic, and cluttered mix-traffic environments. Towards that end, we propose a new safety-enhancement scheme by exploiting vehicle dynamics and multi-step predictions. The goal is to predict any potential collisions over a prediction horizon T_n and correct the unsafe (exploratory) actions accordingly. As we consider a mixed traffic with HDVs, a proper model is needed to predict the high-level decisions of human drivers. In this paper, we use IDM [39] to predict the longitudinal acceleration of the HDVs, based on the current speed and distance headway. In addition, we exploit the MOBIL lane change model [40] to predict the lane-changing behavior of HDVs, which makes a lane-changing decision when it is safe and there is an extra acceleration gain. The high-level decisions of AVs are generated by the MARL agent with the actions defined in Section III-A. These high-level acceleration and lane-change decisions will be realized through low-level PID controllers. The vehicle trajectories are then propagated based on the kinematic bicycle model [41]. We call the high-level decision-induced trajectories motion primitives and show the described framework and simulation setup in Fig. 2.

A. Priority Assignment

With the HDV motion models, one can predict whether a collision can happen in the next T_n steps based on the joint

motion primitives from all AVs. Therefore, it is attempting to use the joint action from all AVs to design the safety-enhancement scheme. However, while it is relatively straightforward to determine whether collisions can happen given a joint action, it is very computationally costly to determine a joint safe action, if a collision is detected, as the action space is $|\mathcal{A}_i|^N$ with N being the number of AVs. It quickly becomes computationally intractable as N grows, especially considering that the considered application has stringent real-time constraints. As such, we propose a sequential, priority-based safety enhancement scheme that has great computation efficiency and is thus suitable for real-time implementations. The principle is that we coordinate the AVs in a sequential order, prioritizing AVs with smaller safety margins. For example, AVs near the end of the merge lane or near the defined safety boundary (e.g. distance headway very close to the defined threshold) should have higher priorities.

More specifically, the following rationales are considered for priority assignments:

- 1) Vehicles on the merge lane should have higher priorities compared to vehicles on the through lane as vehicles on the merge lane face a time-critical merging task.
- 2) Merging vehicles closer to the merge lane end should have higher priorities as they are more probable to cause collisions and deadlocks [7].
- 3) Vehicles with smaller time headway should have a higher priority as they are more likely to collide with the preceding vehicles.

Based on the above rationales, we construct the priority index p_i of the ego vehicle i as follows:

$$p_i = \alpha_1 p_m + \alpha_2 p_d + \alpha_3 p_h + w_i, \quad (10)$$

where α_1 , α_2 and α_3 are positive weighting factors for the merging priority metric p_m , distance-to-end metric p_d , and time headway metric p_h , respectively; and $w_i \sim \mathcal{N}(0, 0.01)$ is a small random variable introduced to avoid two vehicles having the same priority indices. Specifically, p_m is defined as:

$$p_m = \begin{cases} 0.5, & \text{if on merge lane;} \\ 0, & \text{otherwise,} \end{cases} \quad (11)$$

which assigns priority score to vehicles on the merge lane. The distance-to-end priority score p_d is defined as:

$$p_d = \begin{cases} \frac{x}{L}, & \text{if on merge lane;} \\ 0, & \text{otherwise,} \end{cases} \quad (12)$$

where x and L are the distance the ego vehicle has traveled on the connecting ramp and the length of the ramp (see Fig. 1), respectively. Finally, we define the time headway priority score to measure the headway priority as $p_h = -\log \frac{d_{\text{headway}}}{t_h v_t}$, where we use the time headway definition in Eqn. 5 .

B. Priority-based Safety Supervisor

In this subsection, we present the proposed priority-based safety supervisor. Specifically, at each time step t , with the predicted HDV motions and assigned priority scores for all AVs as discussed above, the safety supervisor first generates

a priority list for the AVs, \mathcal{P}_t , with their priority scores in a descendant order, i.e., vehicle with the highest priority is on top of the list. Then the AV on the top of the obtained list, indexed by $\mathcal{P}_t[0]$, is selected for safety check. More specifically, based on the (exploratory) action generated from the action network of vehicle $\mathcal{P}_t[0]$, the safety supervisor will examine whether the motion primitive induced by the exploratory action will conflict with its neighboring vehicles $\mathcal{N}_{\mathcal{P}_t[0]}$ (both AVs and HDVs) in a considered time horizon T_n , where T_n is a hyper-parameter that can be tuned. The motions of HDVs are predicted using the human-driver decision models and vehicle kinematic model discussed above, whereas the motions of all other (lower-priority) AVs are predicted assuming same actions from the last step. As the predicted trajectories of the considered vehicles (i.e., $\mathcal{P}_t[0] \cup \mathcal{N}_{\mathcal{P}_t[0]}$) are all T_n -step sequences, a collision can be detected if any two sequences have a distance below a prescribed threshold at any step k , $k = 1, \dots, T_n$. If no collision is predicted, the exploratory action will be chosen as the actual action for vehicle $\mathcal{P}_t[0]$.

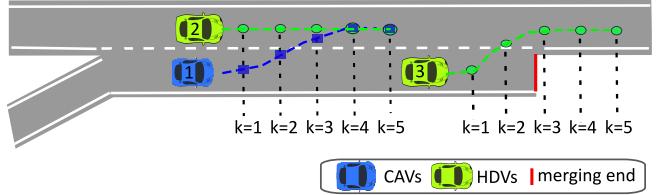


Figure 5: Illustration of trajectory conflict for $T_n = 5$ steps.

On the other hand, if the predicted trajectory of vehicle $\mathcal{P}_t[0]$ is conflicting with other considered vehicles, the exploratory action is determined as unsafe and a new “safe” action will be used to replace the original action. A trajectory conflict is illustrated in Fig. 5, where the motion primitive of vehicle 1 conflicts with the predicted trajectory of vehicle 2 (HDV) at time step 4 and 5. The exploratory action from vehicle 1 is deemed as unsafe. Here we assume no collisions among HDVs (i.e., rational drivers), which can (almost) be guaranteed as the IDM and MOBIL models are extremely safety-focused. Then the safety supervisor enumerate other (valid) candidate actions and pick the best action based on the safety margin as follows:

$$a'_t = \arg \max_{a_t \in \mathcal{A}_{\text{valid}}} \left(\min_{k \in T_n} d_{\text{sm},k} \right). \quad (13)$$

where $\mathcal{A}_{\text{valid}}$ is the set of valid actions at time step t . The safety margin $d_{\text{sm},k}$ at the prediction time step k can be obtained as follows:

- if the action is changing lanes, i.e., *turn left* or *turn right*, the safety margin is defined as the minimum distance to the preceding and the following vehicles on the current and target lanes. An example is shown in the top subfigure of Fig. 6.
- if the action is *speed up*, *idle*, or *slow down*, the safety margin is set as the minimum distance headway. An example is shown in the bottom subfigure of Fig. 6.

After the action of the vehicle $\mathcal{P}_t[0]$ is decided, its trajectory can be (re-)generated. Then vehicle $\mathcal{P}_t[0]$ is removed from the list and the second highest becomes the first, i.e., $\mathcal{P}_t[i] \leftarrow$

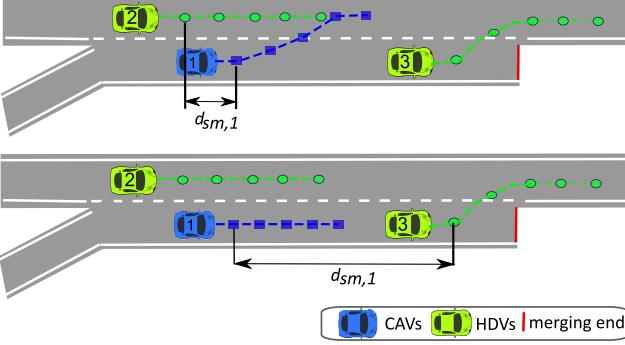


Figure 6: Illustration of safety margin definitions. Top: safety margin if vehicle 1 turns left; and Bottom: safety margin when vehicle 1 keeps straight.

$\mathcal{P}_t[i+1]$, $i = 1, 2, \dots$. Then the same safety-check procedure discussed above is applied to the vehicle corresponding to new $\mathcal{P}_t[0]$, except that when determining collisions, instead of using actions from the last step to generate the trajectories, the motion primitives of the higher-priority vehicles (those who have been processed through the safety check procedure) with the safety-proved actions are used. The procedure will continue until the priority list \mathcal{P}_t is emptied. The details of the proposed priority-based safety supervisor are given in Algorithm 1.

Remark 1. *The priority-based safety supervising scheme can be realized through vehicle-to-infrastructure [48] (V2I) communication where a central communication station near the ramp can observe the HDVs and communicate with the AVs. At each discrete time t , the infrastructure agent will determine the priority scores for the AVs based on the observed traffic. AVs send their exploratory actions to the infrastructure and Algorithm 1 is then performed to generate the updated, safe actions for the AVs. As Algorithm 1 is sequential and thus computationally efficient, with reasonable computation power at the infrastructure, it is expected that the algorithm can be implemented in real-time, i.e., providing updated controls within one sampling time step.*

Remark 2. *The prediction horizon T_n is an important hyperparameter in the safety-enhancement scheme. If T_n is too small, the safety supervisor is “short-sighted” and can lead to no feasible solutions after a few steps. On the other hand, if T_n is too large, the uncertainty of HDVs (the actual vehicle motion in the simulation has noisy perturbations from the human driver models used to predict the trajectories) are propagated and the results tend to be conservative in order to guarantee the safety in a large horizon. In our work, we use cross-validations and find that $T_n = 6$ or 7 is the best choice (see e.g., Fig. 11 and Table I).*

Pseudo-code of the proposed MARL with the priority-based safety supervisor is shown in Algorithm 2. The hyperparameters include: the (time)-discount factor γ , the learning rate η , the total number of training epochs M , the epoch length T , and the coefficients for the loss function β_1 and β_2 . In

Algorithm 1: Priority-based Safety Supervisor

Parameter: $L, \alpha_1, \alpha_2, \alpha_3, t_h, w, T_n$.
Output : $a_i, i \in \mathcal{V}$.

```

1 for  $i = 0$  to  $N$  do
2   compute the priority scores according to Eqn. 10;
3   rearrange ego vehicles to list  $\mathcal{P}_t$  according to their
   priority scores.
4 end
5 for  $j = 0$  to  $|\mathcal{P}_t|$  do
6   obtain the highest-priority vehicle  $\mathcal{P}_t[0]$ ;
7   find its neighboring vehicles  $\mathcal{N}_{\mathcal{P}_t[0]}$ ;
8   predict trajectories  $\zeta_v, v \in \mathcal{P}_t[0] \cup \mathcal{N}_{\mathcal{P}_t[0]}$  for  $T_n$ 
   time steps.
9   if trajectories are overlapped then
10    replace the risky action as  $a_t \leftarrow a'_t$  according
        to Eqn. 13;
11    replace the trajectory  $\zeta_{\mathcal{P}_t[0]}$  with  $\zeta'_{\mathcal{P}_t[0]}$ 
12 end
13 remove  $\mathcal{P}_t[0]$  from  $\mathcal{P}_t$ ;
14 update  $\mathcal{P}_t[i] \leftarrow \mathcal{P}_t[i+1], i = 1, 2, \dots$ 
15 end

```

each epoch, each agent collects the state information and samples actions by applying the action masking strategy to avoid invalid actions (Lines 4–7). Then the exploratory actions from the MARL will be checked by the priority-based safety supervisor detailed in Algorithm 1 (Line 9). If the action is unsafe, then the safety supervisor will replace the risky action with a safe action according to Eqn. 13. The safe action will be taken by the agent and the corresponding experience will be collected and saved to the replay buffer (Lines 10–17). The parameters of the policy network are updated using the collected experience sampled from the on-policy experience buffer after the completion of each episode (Lines 20–26). The DONE signal is flagged if either the episode is completed or a collision occurs. After receiving the DONE flag, all agents are reset to their initial states to start a new epoch (Lines 28).

V. NUMERICAL EXPERIMENTS

In this section, we evaluate the performance of the proposed MARL algorithm in terms of training efficiency and collision rate in the on-ramp merging scenario illustrated in Fig. 1. The length of the road is 520 m, where the entrance of the merge lane is at 320 m and the length of the merge lane is $L = 100$ m. There are 12 spawn points evenly distributed on the through lane and the ramp lane from 0 m to 220 m. The vehicles exceeding the road will be removed from displaying while the kinematics are still updated. Specifically, we consider three levels of traffic densities with different number of initial vehicles defined as:

- *Easy mode:* 1-3 AVs and 1-3 HDVs.
- *Medium mode:* 2-4 AVs + 2-4 HDVs.
- *Hard mode:* 4-6 AVs + 3-5 HDVs.

In each training episode, a different number of HDVs and AVs will randomly appear at the spawn points with a random position noise (uniformly distributed in [-1.5m, 1.5m]) added to each initial spawn position. The initial speed is randomly chosen between 27 to 29 m/s. The vehicle control sampling

Algorithm 2: MARL for AVs with Safety Supervisor

Parameter: $\gamma, \eta, T, M, \beta_1, \beta_2$.
Output : θ .

```

1 initialize  $s_0, t \leftarrow 0, \mathcal{D} \leftarrow \emptyset$ ;
2 for  $j = 0$  to  $M - 1$  do
3   for  $t = 0$  to  $T - 1$  do
4     for  $i \in \mathcal{V}$  do
5       observe  $s_i$ ;
6       update  $a_{i,t} \sim \pi_{\theta_i}(\cdot | s_i)$  with action masking.
7     end
8     for  $i \in \mathcal{V}$  do
9       check the actions by Algorithm 1;
10      if safe then
11        execute  $a_{i,t}$ ;
12        update  $\mathcal{D}_i \leftarrow (s_{i,t}, a_{i,t}, r_{i,t}, v_{i,t})$ 
13      end
14      else
15        update  $a_{i,t} \leftarrow a'_{i,t}$  and execute  $a'_{i,t}$ ;
16        update  $\mathcal{D}_i \leftarrow (s_{i,t}, a'_{i,t}, r_{i,t}, v_{i,t})$ .
17      end
18    end
19    update  $t \leftarrow t + 1$ 
20    if DONE then
21      for  $i \in \mathcal{V}$  do
22        update  $\theta_i \leftarrow \theta_i + \eta \nabla_{\theta_i} J(\theta_i)$ 
23      end
24    end
25    initialize  $\mathcal{D}_i \leftarrow \emptyset, i \in \mathcal{V}$ ;
26    update  $j \leftarrow j + 1$ 
27  end
28  update  $s_0, t \leftarrow 0$ 
29 end

```

frequency is 5 Hz, i.e., AVs take an action every 0.2 seconds. A 5% random noise is added to the predicted acceleration and steering angle for HDVs. We train all MARL algorithms over 2 million steps with 3 different random seeds while the same random seeds are shared among the agents, which is around 20,000 episodes with episode horizon $T = 100$ steps. We evaluate the algorithm over 3 episodes every 200 training episodes. We set $\gamma = 0.99$ and the learning rate $\eta = 5e^{-4}$; The coefficients w_c, w_s, w_h , and w_m for the reward function are set as 200, 1, 4, and 4, respectively. The priority coefficients α_1, α_2 and α_3 are equally set as 1. The weighting coefficients β_1 and β_2 for the loss function are chosen as 1 and 0.01, respectively. Here we call the MARL algorithm, without the safety supervisor, proposed in Section III as the baseline method.

The simulation environment is modified from the gym-based highway-env simulator [49] and is open-sourced². We use the default parameters of the IDM and MOBIL models which can be found as in the highway-env simulator [49]. The experiments have been performed in a Ubuntu 18.04 server with AMD 9820X processor and 64 GB memory. The video demo of the training process can be found at the site³.

A. Network and Reward Function Designs

In this subsection, we will evaluate the performance of the proposed MARL framework under different network de-

signs, i.e., shared (baseline) v.s. separate actor-critic networks (baseline without sharing), different reward function designs, and local (baseline) v.s. global rewards (baseline with global reward). Note that using separate actor and critic networks requires independent minimization of the value loss and the policy loss as in [25].

Fig. 7 shows the evaluation performance during training for the comparison between sharing and not sharing actor-critic network parameters. It is clear that sharing actor-critic network parameters consistently achieves higher reward and lower variance (characterized by smaller shaded regions) in all three traffic levels. This is due to the fact that if the actor and critic networks are separate, the actor may not be able to learn meaningful information until the critic network has been well trained, which may take long training time. On the other hand, by sharing the parameters, the actor network can benefit from the learned state representation via the critic network [50]. Therefore, we adopt the network-sharing strategy in our network design.

We next investigate the proposed local reward function by comparing it with the global reward design used in [18], [21], where the reward of the i th agent at time step t is the averaged global reward $r_{i,t} = \frac{1}{N} \sum_{j=1}^N r_{j,t}$. Fig. 8 shows the performance comparison between the proposed local reward design and the global reward design (with shared actor-critic parameters). As expected, the proposed local reward outperforms the global reward design in terms of higher evaluation rewards as well as lower variance across all three traffic scenarios. In the *Easy* mode, the global reward design performs well and achieves a reasonable reward due to the small number of AVs, while it fails the control tasks in the *Medium* and *Hard* modes as it suffers from the credit assignment issues [44] and the fact that the assigned average global rewards have less correlation with individual agent's actions as the number of agents increases.

B. Curriculum Learning

In this subsection, we adopt curriculum learning [51] to speed up the learning and improve the performance for the *Hard* mode. Instead of learning the *Hard* mode directly, we build upon the trained model from the easier modes (i.e., *easy* and *medium*) and train the models to achieve higher efficiency. Curriculum learning is especially advantageous for safety-critical tasks (e.g. autonomous driving) as starting from a decent model can greatly reduce the number of “blind” explorations that can be risky.

Fig. 9 shows evaluation performance comparison between the baseline method (i.e., starting from scratch) and curriculum learning (baseline + curriculum learning) for the *Hard* traffic mode. It is obvious that learning based on the trained model from easier tasks greatly expedites the speed of convergence and improves the final model performance. The average speed during the training, as shown in Fig. 10, indicates that the curriculum learning strategy also improves the average vehicle speed, thus achieving high throughput. Therefore, we apply the curriculum learning in the following experiments for *Hard* traffic modes.

²See https://github.com/Derekabc/MARL_CAVs

³See <https://drive.google.com/drive/folders/1437My4sDoyPFsUjrThmlu1oJjTkTkvJ7?usp=sharing>

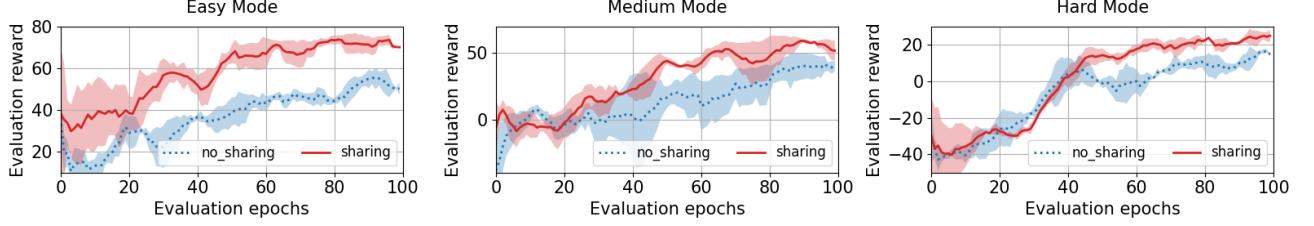


Figure 7: Evaluation curves during training with and without “sharing” actor-critic network for different traffic levels. The best learning curve is in bold. The shaded region denotes the standard deviation over 3 random seeds.

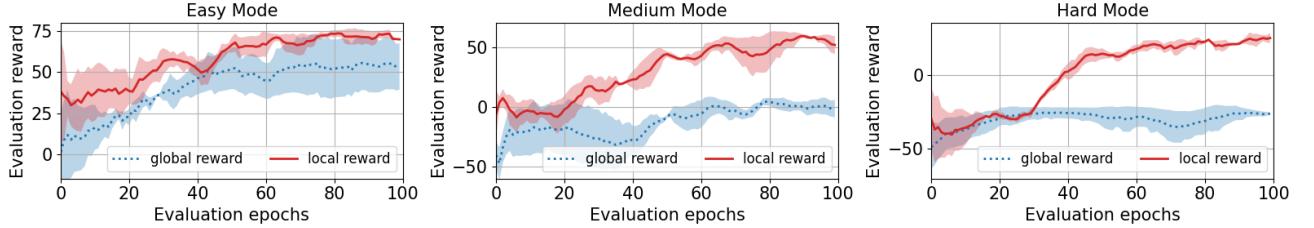


Figure 8: Evaluation curves during training with different reward functions for different traffic levels.

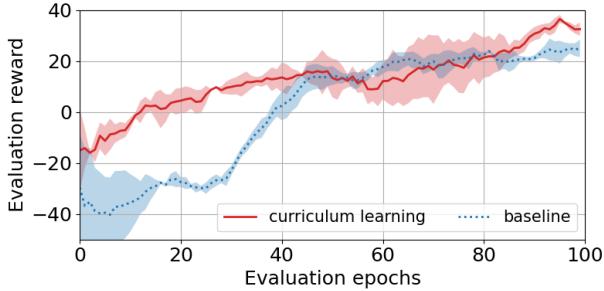


Figure 9: Evaluation curves during training with or without curriculum learning for *Hard* traffic mode.

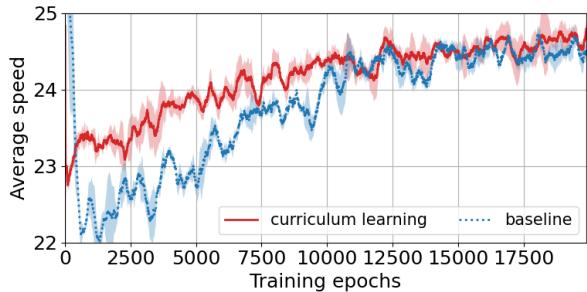


Figure 10: Average speed during training with and without curriculum learning for *Hard* traffic mode.

C. Performance of the Priority-based Safety Supervisor

In this subsection, we evaluate the effectiveness of the proposed priority-based safety supervisor. As shown in Fig. 11, the proposed priority-based safety supervisor method has a better sample efficiency, evidenced by faster converge speed in all three traffic densities. This is because most unsafe actions are replaced with safe ones by the safety supervisor, especially in the earlier exploration phase, which avoids early

terminations and thus improves learning efficiency.

Fig. 12 shows the average vehicle speed during the training, which is an indication of traffic throughput. It is clear that the algorithms with the safety supervisor maintain higher training speed than the baseline method (i.e., without safety supervisor). This shows that the proposed safety supervisor is not only beneficial for training but also leads to better traffic efficiency. It can also be seen that vehicle speeds are slower as the traffic density increases, which is reasonable as the interactions are more frequent in a dense traffic and lower speed is safer to avoid collisions.

After training, MARL algorithms for each traffic density are tested over 3 random seeds for 30 epochs and the average collision rates and vehicle speeds are shown in Table I. We can see that with the safety supervisor, the MARL can run without collisions in the *Easy* mode, while the baseline method has a collision rate of 0.03. For the more challenging *Medium* and *Hard* modes, MARL with safety supervisor also significantly outperforms the baseline method with much lower collision rates. The average speed also indicates that the safety supervisor leads to higher traffic throughput. Note that the proposed method with the safety supervisor tends to use a slightly low average speed (e.g., baseline + $T_n = 6$ (24.38 m/s) v.s. baseline (24.60 m/s)) in the *Hard* mode during testing to keep a low collision rate (e.g., baseline + $T_n = 6$ (0.04) v.s. baseline (0.20)), while the speed during training is always higher than the baseline method in all traffic levels.

D. Comparison with State-of-the-art Benchmarks

In this subsection, we compare the proposed method with several state-of-the-art MARL benchmarks, MAA2C, MAPPO and MAACKTR, as mentioned in Section II. All the benchmarks are implemented by sharing parameters among agents to deal with dynamic numbers of agents, and using the global reward and the discrete action space. We adopt the 6-step

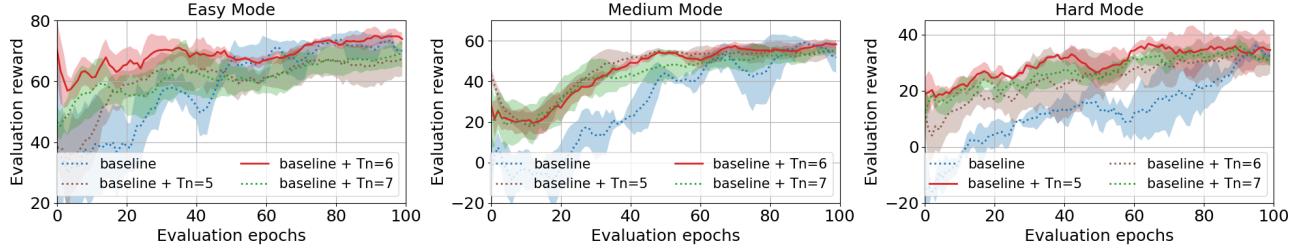


Figure 11: Evaluation curves for the n-step priority-based safety supervisor.

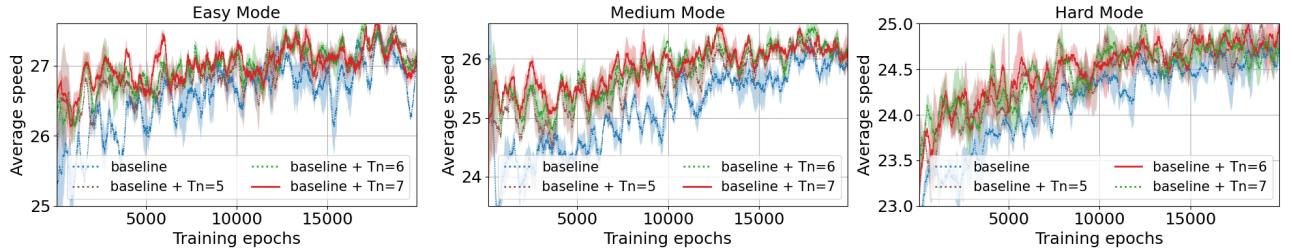


Figure 12: Average speed during training for the n-step priority-based safety supervisor.

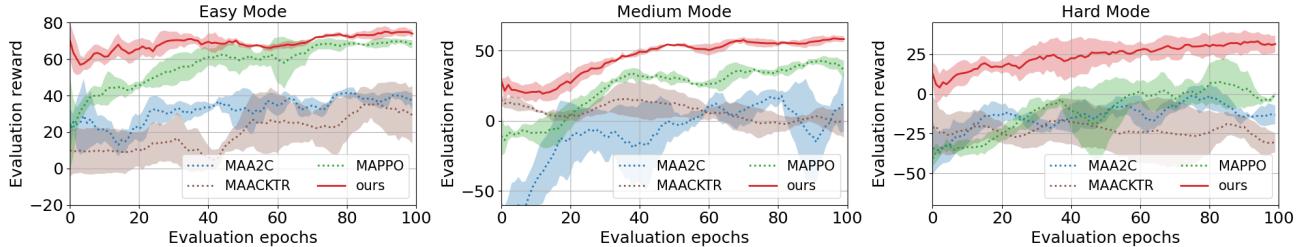
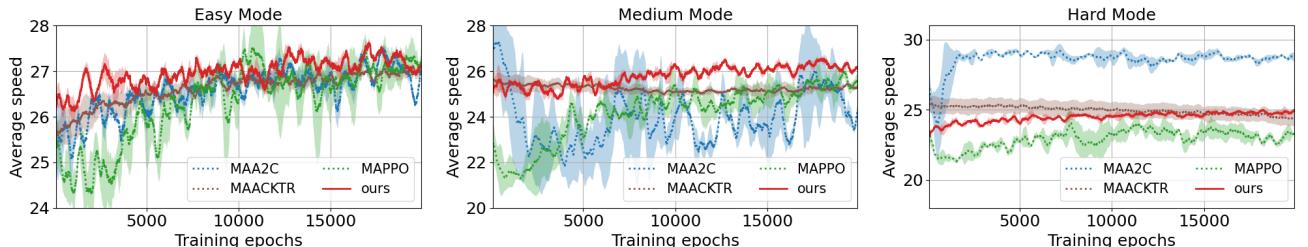
Figure 13: Evaluation curves comparison between the proposed MARL policy (ours: baseline + $T_n = 6$) and 3 state-of-the-art MARL benchmarks.Figure 14: Average speed comparison between the proposed MARL policy (ours: baseline + $T_n = 6$) and 3 state-of-the-art MARL benchmarks.

Table I: Testing performance comparison of collision rate and average speed between n-step safety supervisor. The best values are in bold.

Scenarios	Metrics	baseline	baseline + Tn=5	baseline + Tn=6	baseline + Tn=7
Easy Mode	collision rate	0.03	0	0	0
	avg. speed [m/s]	27.19	27.15	27.37	27.15
Medium Mode	collision rate	0.08	0.03	0.02	0.03
	avg. speed [m/s]	26.07	26.00	26.17	26.17
Hard Mode	collision rate	0.20	0.05	0.04	0.04
	avg. speed [m/s]	24.60	24.47	24.38	24.50

safety supervisor (ours: baseline + $T_n = 6$) in all traffic levels as it takes a good trade-off between collision rate and

prediction efficiency as shown in Section V-C.

Fig. 13 shows the evaluation results during training for all the MARL algorithms. The proposed method (ours) consistently outperforms the benchmarks in all traffic levels. The proposed method shows even greater advantage in terms of sample efficiency and training performance in the *Hard* mode over the benchmarks. Fig. 14 shows the proposed method has relatively higher average training speed which leads to high training efficiency. Note that it is not wise to have high vehicle speed as in MAA2C and MAACKTR methods in dense traffic, which will lead to very high collision rates as shown in Table II. The testing results in Table II also show

that the proposed method has lower collision rates and higher efficiency than other benchmark algorithms.

Table II: Testing performance comparison of collision rate and average speed between the proposed method and 3 state-of-the-art benchmarks.

Scenarios	Metrics	MAA2C	MAACKTR	MAPPO	ours
Easy Mode	collision rate	0.08	0.09	0.01	0
	avg. speed [m/s]	25.43	26.43	27.36	27.37
Medium Mode	collision rate	0.14	0.12	0.04	0.02
	avg. speed [m/s]	25.40	25.22	25.66	26.17
Hard Mode	collision rate	0.97	0.50	0.16	0.04
	avg. speed [m/s]	29.02	25.37	24.22	24.38

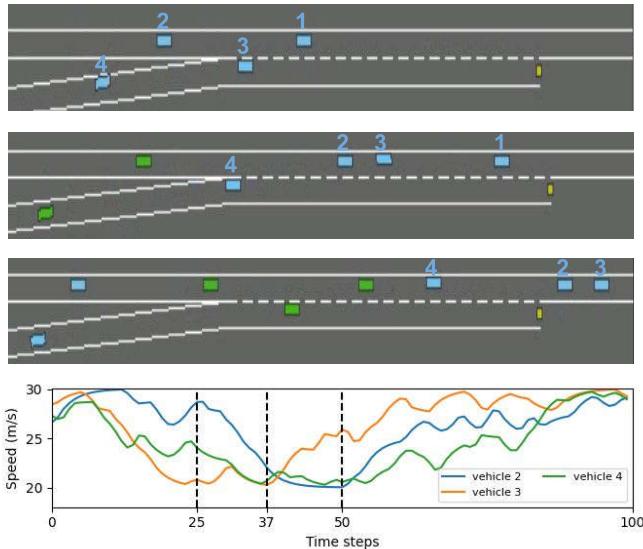


Figure 15: Frames show the learned policy. Below figure shows the corresponding speed of the AVs.

E. Policy Interpretation

In this subsection, we attempt to interpret the learned AV behaviors. As an example, Fig. 15 shows the snapshots at time steps 25, 37, and 50, as well as the speeds of agents 2-4. It can be observed that at the time step 25, vehicle 2 starts to slow down and makes space for vehicle 3 to merge. Vehicle 3 accelerates to merge while keeping an adequate distance headway with vehicle 1. Then vehicle 3 successfully merges into the through lane and starts to speed up at time step 37. At the same time, vehicle 2 still keeps a low speed to keep a safe headway distance to vehicle 3. At time 50, vehicle 2 speeds up while keeping a lower speed than vehicle 3 to maintain a safe distance headway. Similar patterns are also observed in vehicle 4.

VI. CONCLUSIONS

In this paper, we formulated the problem of on-ramp merging in a mixed-traffic as an on-policy MARL, and we developed an efficient MARL algorithm featuring action masking, local reward design, curriculum learning, and critic-actor parameter sharing. A novel priority-based safety supervisor

was also developed to enhance safety, improve learning efficiency, and increase traffic throughput. Comprehensive experiments were conducted to compare with several state-of-the-art algorithms, which showed that the proposed approach consistently outperformed the benchmark approaches in terms of training efficiency and collision rate. In future works, we will incorporate more comprehensive human driver models so that HDV motions can be more accurately predicted. We will also develop a more realistic simulation environment by incorporating data from real-world traffic systems.

REFERENCES

- [1] “Future of driving,” <https://www.tesla.com/autopilot>, accessed: 2021-03-31.
- [2] “Apollo open platform,” <https://apollo.auto/developer.html>, accessed: 2021-03-31.
- [3] V. V. Dixit, S. Chand, and D. J. Nair, “Autonomous vehicles: disengagements, accidents and reaction times,” *PLoS one*, vol. 11, no. 12, p. e0168054, 2016.
- [4] F. M. Favaro, N. Nader, S. O. Eurich, M. Tripp, and N. Varadaraju, “Examining accident reports involving autonomous vehicles in california,” *PLoS one*, vol. 12, no. 9, p. e0184952, 2017.
- [5] D. Ni and J. D. Leonard II, “A simplified kinematic wave model at a merge bottleneck,” *Applied mathematical modelling*, vol. 29, no. 11, pp. 1054–1072, 2005.
- [6] L. Leclercq, J. A. Laval, and N. Chiabaut, “Capacity drops at merges: An endogenous model,” *Procedia-Social and Behavioral Sciences*, vol. 17, pp. 12–26, 2011.
- [7] 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.
- [8] J. Rios-Torres and A. A. Malikopoulos, “A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1066–1077, 2016.
- [9] L. N. Jacobson, K. C. Henry, and O. Mehyar, *Real-time metering algorithm for centralized control*, 1989, no. 1232.
- [10] J. Hourdakis and P. G. Michalopoulos, “Evaluation of ramp control effectiveness in two twin cities freeways,” *Transportation Research Record*, vol. 1811, no. 1, pp. 21–29, 2002.
- [11] Y. Lin, J. McPhee, and N. L. Azad, “Anti-jerk on-ramp merging using deep reinforcement learning,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 7–14.
- [12] W. Cao, M. Mukai, T. Kawabe, H. Nishira, and N. Fujiki, “Cooperative vehicle path generation during merging using model predictive control with real-time optimization,” *Control Engineering Practice*, vol. 34, pp. 98–105, 2015.
- [13] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.
- [14] J. Lubars, H. Gupta, A. Raja, R. Srikant, L. Li, and X. Wu, “Combining reinforcement learning with model predictive control for on-ramp merging,” *arXiv preprint arXiv:2011.08484*, 2020.
- [15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [16] J. Wang, T. Shi, Y. Wu, L. Miranda-Moreno, and L. Sun, “Multi-agent graph reinforcement learning for connected automated driving.”
- [17] P. Palanisamy, “Multi-agent connected autonomous driving using deep reinforcement learning,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.
- [18] M. Kaushik, K. M. Krishna *et al.*, “Parameter sharing reinforcement learning architecture for multi agent driving behaviors,” *arXiv preprint arXiv:1811.07214*, 2018.
- [19] P. Y. J. Ha, S. Chen, J. Dong, R. Du, Y. Li, and S. Labi, “Leveraging the capabilities of connected and autonomous vehicles and multi-agent reinforcement learning to mitigate highway bottleneck congestion,” *arXiv preprint arXiv:2010.05436*, 2020.
- [20] S. Bhalla, S. G. Subramanian, and M. Crowley, “Deep multi agent reinforcement learning for autonomous driving,” in *Canadian Conference on Artificial Intelligence*. Springer, 2020, pp. 67–78.

- [21] 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.
- [22] C. Yu, X. Wang, X. Xu, M. Zhang, H. Ge, J. Ren, L. Sun, B. Chen, and G. Tan, "Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 735–748, 2019.
- [23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.
- [24] 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.
- [25] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.
- [26] OpenAI, ;, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019.
- [27] N. Naderializadeh, J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Transactions on Wireless Communications*, 2021.
- [28] D. Chen, Z. Li, T. Chu, R. Yao, F. Qiu, and K. Lin, "Powernet: Multi-agent deep reinforcement learning for scalable powergrid control," *arXiv preprint arXiv:2011.12354*, 2020.
- [29] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.
- [30] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.
- [31] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1774–1783.
- [32] J. K. Terry, N. Grammel, A. Hari, L. Santos, and B. Black, "Revisiting parameter sharing in multi-agent deep reinforcement learning," 2021.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [34] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *arXiv preprint arXiv:1708.05144*, 2017.
- [35] I. Elsayed-Aly, S. Bharadwaj, C. Amato, R. Ehlers, U. Topcu, and L. Feng, "Safe multi-agent reinforcement learning via shielding," *arXiv preprint arXiv:2101.11196*, 2021.
- [36] Z. Cai, H. Cao, W. Lu, L. Zhang, and H. Xiong, "Safe multi-agent reinforcement learning through decentralized multiple control barrier functions," *arXiv preprint arXiv:2103.12553*, 2021.
- [37] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.
- [38] T. Ayres, L. Li, D. Schleuning, and D. Young, "Preferred time-headway of highway drivers," in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*. IEEE, 2001, pp. 826–829.
- [39] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.
- [40] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.
- [41] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 812–818.
- [42] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," in *Modeling complexity in economic and social systems*. World Scientific, 2002, pp. 355–369.
- [43] J. Wang, Y. Zhang, T.-K. Kim, and Y. Gu, "Shapley q-value: A local reward approach to solve global reward games," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7285–7292.
- [44] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [45] D. Bagnell and A. Ng, "On local rewards and scaling distributed reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 18, pp. 91–98, 2005.
- [46] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.
- [47] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.
- [48] C.-M. Chou, C.-Y. Li, W.-M. Chien, and K.-c. Lan, "A feasibility study on vehicle-to-infrastructure communication: Wifi vs. wimax," in *2009 tenth international conference on mobile data management: systems, services and middleware*. IEEE, 2009, pp. 397–398.
- [49] E. Leurent, "An environment for autonomous driving decision-making," <https://github.com/eleurent/highway-env>, 2018.
- [50] L. Graesser and W. L. Keng, *Foundations of deep reinforcement learning: theory and practice in Python*. Addison-Wesley Professional, 2019.
- [51] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *2018 ieee intelligent vehicles symposium (iv)*. IEEE, 2018, pp. 1885–1890.