Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction

Hanna Krasowski*, Xiao Wang*, and Matthias Althoff

***Abstract*— Machine learning approaches often lack safety guarantees, which are often a key requirement in real-world tasks. This paper addresses the lack of safety guarantees by extending reinforcement learning with a safety layer that restricts the action space to the subspace of safe actions. We demonstrate the proposed approach using lane changing in autonomous driving. To distinguish safe actions from unsafe ones, we compare planned motions with the set of possible occupancies of traffic participants generated by set-based predictions. In situations where no safe action exists, a verified fail-safe controller is executed. We used real-world highway traffic data to train and test the proposed approach. The evaluation result shows that the proposed approach trains agents that do not cause collisions during training and deployment.**

I. Introduction

Self-driving techniques have the potential to improve mobility in terms of safety and traffic efficiency. One of the most crucial tasks for autonomous vehicles is to plan their motion through traffic without harming other traffic participants. The recent development of motion planning techniques has become more data driven due to the advancement in computation power and the amount of available traffic data. Compared to rule-based methods, data-driven approaches require much less expert knowledge based on the ability to learn complex dependencies from data. Motion planning tasks can be modeled as Markov decision processes, for which reinforcement learning (RL) provides potential solutions. RL's core idea is that an agent learns to interact with the environment by exploring different actions and receiving the next state of the environment and a reward. The exploration process of RL impedes its applicability to real-world problems since unsafe actions are possibly executed.

Various approaches have been proposed to increase the safety of RL methods by modifying the optimality criterion [1], [2] or by verifying the exploration processes with external guidance [3]–[10]. By modifying the optimality objective, agents behave more cautious than those trained without a risk measure included in the objective; however, the absence of unsafe behaviors cannot be proven. In contrast, by verifying the safety of the action and excluding possible unsafe actions, we can ensure that the exploration process is safe. Therefore, we focus on verifying the safety of proposed actions and ensuring safety if the agent fails to find a safe action.

* The first two authors have contributed equally to this work.

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
`hanna.krasowski@tum.de, xiao.wang@tum.de, althoff@in.tum.de`

In this paper, we propose a safe RL framework for motion planning based on our previous work on autonomous lane changing [6]. Our contributions are threefold:

1) We benchmark state-of-the-art model-free RL algorithms to solve high-level behavior planning problems for highway driving.
2) We propose a framework to integrate RL methods in our developed safety layer for autonomous vehicles.
3) We evaluate the proposed approach using a real-world highway traffic dataset.

The remainder of this paper is organized as follows: Section II provides an overview of recent developments in safe RL and safe motion planning techniques. Section III introduces individual modules of our safe RL framework for motion planning. In Section IV, we evaluate the proposed method in real-world highway scenarios. Section V gives the conclusion.

II. Related Work

Safe RL approaches are distinguished in [11] by approaches that modify the optimization criterion and by approaches that modify the exploration process. As previously discussed, only approaches modifying exploration are verifiably safe; thus, we focus on this technique in the subsequent literature review.

A. Modification of Exploration Process

One method to alter the action selection process is to prioritize actions that are estimated to be safer [3]. However, this approach does not prove the nonexistence of unsafe behaviors. Another approach starts with a verified agent model and updates the agent only if the safety requirements are preserved [4], [5]. However, a verified agent model is not always available for complex tasks. A third alternative is to verify which actions are safe and to restrict the action space to safe actions [6]–[10]. However, if all actions are verified as unsafe, safety is no longer guaranteed. To guarantee safety, using the third method, we added a verified fail-safe planner, which holds available a safe action that is activated when the agent fails to identify a safe action.

B. Safety Verification for Autonomous Vehicles

Researchers have proposed various approaches to verify the safety of motion planners for autonomous vehicles. A common method is to predict the most likely motion of other traffic participants [12] or a probability distribution of their future behaviors [13]. The planned trajectories are executed if they do not collide with a traffic participant according to its prediction. The limitation of this method is that collisions still happen if other traffic participants' behavior deviates from their prediction. In another approach, a minimum requirement for safe motion planning is that inevitable collision states are avoided [14]. A system is in an inevitable collision state if it collides with other traffic participants irrespective of the action taken. However, the calculation of inevitable collision states suffers from the curse of dimensionality. Another possibility is to apply logical reasoning, which uses deduction to prove correct behavior based on given rules [4], [5], [8]. However, logical reasoning is typically not appropriate for online verification, which is required in this work. Furthermore, logical reasoning requires human intervention.

Reachability analysis verifies the safety of planned trajectories by computing all possible future motions of obstacles and checking whether they intersect with the occupancy of the ego vehicle [15]. Since computing the exact reachable sets of nonlinear systems is impossible, reachable sets are over-approximated to ensure safety.

III. Reinforcement Learning with Safety Verification

A. Framework

We build a safe RL framework to tackle the safe lane-changing task by integrating a safety layer between the agent and the environment, as shown in Fig. 1. The safety layer guides the exploration process by restricting the action space to the safe subspace of actions. The task of the agent is to reach a goal area on a multilane highway safely. We define an agent's behavior as safe if it does not cause collisions with other traffic participants.

The safety layer receives the current state of the ego vehicle s_{ego} and the states $s_{\text{obstacles}}$ of surrounding obstacles. Using these states, we predict the possible occupancy areas of the surrounding obstacles. We generate trajectories from high-level actions to check for collisions with the predicted occupancies, thereby determining which high-level actions are safe, as described in Section III-C.

The safe action mask generated by the safety layer restricts the agent's actions to safe actions only, as presented in Section III-E.

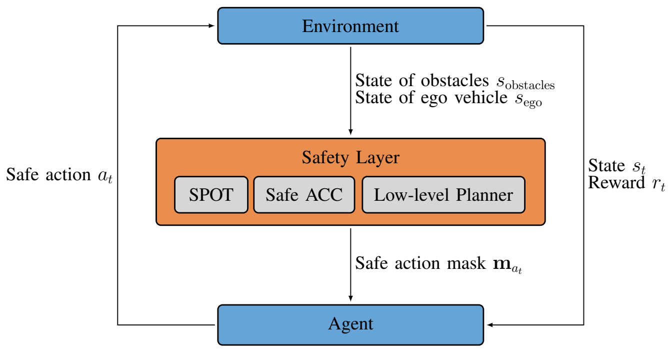
Environment
State of obstacles $s_{\text{obstacles}}$
State of ego vehicle s_{ego}
Safe action a_t
Safety Layer
SPOT
Safe ACC
Low-level Planner
State s_t
Reward r_t
Safe action mask $\mathbf{m}_{a_t}$
Agent

Fig. 1. Reinforcement learning with the safety layer.

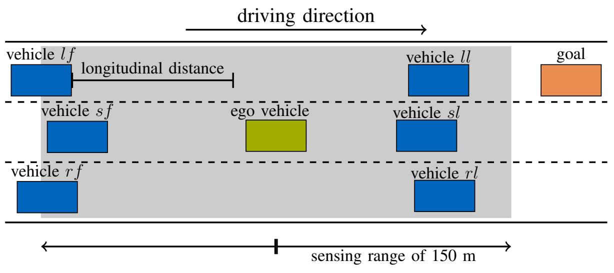

Fig. 2. Schematic representation of a three-lane road with the ego vehicle, surrounding vehicles ij, and the goal area. The gray area depicts the sensing field of the ego vehicle. The obstacle's lane is specified relative to the lane of the ego vehicle by i, i.e., l for the left lane, s for the same lane, and r for the right lane. The relative position of the surrounding vehicle to the ego vehicle is described by j, i.e., l for leading and f for following.

TABLE I

16-Dimensional Continuous State Space

Dim.	State	Description
1-6	d_{ij}	The longitudinal distance of surrounding vehicle ij to the ego vehicle (Fig. 2)
7-12	v_{ij}	The relative velocity of surrounding vehicle ij to the ego vehicle
13	v_{ego}	Absolute velocity of the ego vehicle
14	a_{ego}	Absolute acceleration of the ego vehicle
15	d_{long}	Longitudinal distance from ego vehicle to the goal area
16	d_{lat}	lateral distance from ego vehicle to the goal area

B. Markov Decision Process for High-Level Planning

The discrete action space contains the high-level actions for lane-changing decisions: changing to the left lane, changing to the right lane, continuing in the current lane, and staying in the current lane by activating a safe adaptive cruise control (ACC) [16]. The safe ACC is only activated for fail-safe maneuvers since it ensures safety for an infinite time horizon.

The 16-dimensional continuous state space is shown in Tab. I. The agent is provided with the distance to the goal area, as well as the state of the ego vehicle and the surrounding vehicles, to reach a goal area on a highway safely. We consider the relative velocity and longitudinal distance of six surrounding vehicles in a sensing range of 150 m, as illustrated in Fig. 2. Binary variables are introduced below for further derivations:

- $\mathbf{1}_{\text{reach\_goal}} = 1$ when the ego vehicle reaches the goal area.
- $\mathbf{1}_{\text{goal\_lane}} = 1$ when the ego vehicle drives in the lane of the goal area.
- $\mathbf{1}_{\text{collision}} = 1$ if the ego vehicle collides with other vehicles.
- $\mathbf{1}_{\text{safe\_violation}} = 1$ if the ego vehicle violates the safe distance to the leading vehicle or during a lane change to the following vehicle.

We terminate an episode if the time horizon of the current traffic scenario is reached, the goal area is reached, or the ego vehicle collides with another vehicle. The reward function is defined as

$$r = r_{\text{reach\_goal}} + r_{\text{goal\_lane}} + r_{\text{closer}} + r_{\text{crash}} + r_{\text{safe\_dist}}, \quad (1)$$

where each term is further specified as

$$r_{\text{reach\_goal}} = 100 \cdot \mathbf{1}_{\text{reach\_goal}} \tag{2a}$$

$$r_{\text{goal\_lane}} = 5 \cdot \mathbf{1}_{\text{goal\_lane}} \tag{2b}$$

$$r_{\text{closer}} = d_{\text{long}}(t-1) - d_{\text{long}}(t) \tag{2c}$$

$$r_{\text{crash}} = -100 \cdot \mathbf{1}_{\text{collision}} \tag{2d}$$

$$r_{\text{safe\_dist}} = -10 \cdot (\frac{d_{\text{safe}}}{d_{ij}} - 1) \cdot \mathbf{1}_{\text{safe\_violation}}. \tag{2e}$$

The sparse rewards $r_{\text{reach\_goal}}$ and r_{crash} encourage goal-reaching or collision avoidance behaviors. Additionally, the positive rewards $r_{\text{goal\_lane}}$ and r_{closer} are provided if the ego vehicle gets closer to the goal area in a lateral or longitudinal direction. The penalty $r_{\text{safe\_dist}}$ encourages the agent to keep a safe distance. The safe distance [17] between two vehicles is calculated by

$$d_{\text{safe}} = \frac{1}{2a_{\max}} (v_f^2 - v_l^2) + v_f\, t_{\text{react}}, \tag{3}$$

where v_l and v_f are the the leading and following vehicles' current velocity, respectively, $t_{\text{react}} = 0.32\,\text{s}$ is the reaction time and $a_{\max} = 11.5\,\text{m/s}^2$ is the maximum deceleration of the vehicles. The value for the reaction time is taken from [17]. The maximum deceleration is based on common values for midsize cars like the BMW 320i [18].

C. *Safety Layer*

To check whether a high-level action might result in a collision, we compute the future occupancy of the ego vehicle and that of other traffic participants. If both occupancy sets do not intersect for all consecutive time intervals within a predefined time horizon and if the ego vehicle reaches an invariably safe set [19], a collision is impossible. Figure 3 shows an example of a traffic situation with trajectories and occupancies of two surrounding vehicles. If the occupancy of the ego vehicle intersects with the obstacles' occupancy at a certain time step, e.g., as shown in Fig. 3 c), the corresponding high-level action is regarded as unsafe. The occupancies of the surrounding traffic participants are obtained by using our tool SPOT [20]. SPOT considers the physical limits of surrounding traffic participants and constraints implied by traffic rules, e.g., vehicles are not allowed to drive backward on a highway.

The occupancy of the ego vehicle from high-level actions is obtained by a motion planner and the road network structure. For the go-straight action, the ego vehicle follows the center of its current lane. For lane-changing actions, we assume a duration of 2 s. The precise movement is obtained by a sampling-based trajectory planner [21], which requires the total time t_{total}, the final velocity of the trajectory v_{final}, and the lateral deviation from the given reference path $d_{\text{lat\_ref}}$. The intervals from which our planner samples are defined as

- $t_{\text{total}} \in [0.2\,\text{s}, t_{\max}]$,
- $v_{\text{final}} \in \left[v_{\min}, \max\left(v_{\min}, v_{\text{desired}} + 0.25\, t_{\max} a_{\max}\right)\right]$, where $v_{\min} = \max\left(0\,\text{m/s},\, v_{\text{desired}} - 0.125\, t_{\max} a_{\max}\right)$,
- $d_{\text{lat\_ref}} \in [-2\,\text{m}, 2\,\text{m}]$.

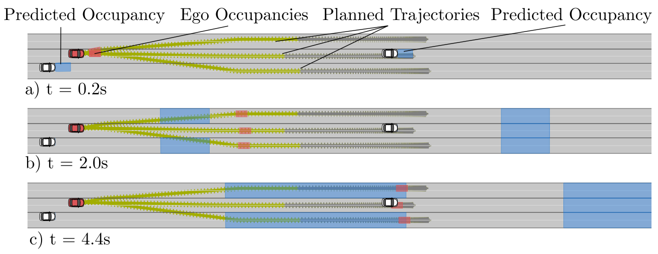

Fig. 3. Example situation for safety verification using set-based prediction [20] and sampling-based trajectory planning [21] with two traffic participants for three time steps. Blue polygons are the predicted occupancies of the surrounding vehicles at specified times. Green lines are the feasible trajectories of the ego vehicle and the gray lines are the appended braking trajectories. The red rectangles symbolize the occupancies of the ego vehicle at the specified time steps.

The desired velocity v_{desired} is defined by the mean velocity necessary for reaching the goal from the initial state in the considered dataset. The planning horizon $t_{\max}$ is set to 2.7 s for go-straight and lane-changing trajectories.

We first exclude meaningless high-level actions that would result in leaving the road, e.g., we exclude changing to the left when driving in the leftmost lane. For each combination of the sampling parameters, a trajectory is generated and checked with the vehicle's kinematic constraints. An optimal trajectory is selected according to the cost function in [21] after the exclusion of kinematically infeasible trajectories. Note that the proposed method can be extended to a continuous action space by converting a sequence of continuous actions to trajectories. We append a braking trajectory with maximum deceleration to the sampled trajectory (cf. Fig. 3). The ego vehicle never follows this braking trajectory but it is utilized to check if the vehicle is in an invariably safe state at the end of its driving trajectory.

If none of the calculated trajectories are considered safe, the fail-safe plan is executed. We utilize the safe ACC from [16] as a fail-safe planner to ensure safety beyond the planning horizon.

D. *Selection of the Reinforcement Learning Algorithm*

Before integrating the action masking in our policy model, we select the RL algorithm for the goal-reaching task on highways. We benchmark three state-of-the-art RL algorithms with a discrete action space, namely deep Q-network (DQN) [22], actor-critic with experience replay (ACER) [23], and proximal policy optimization (PPO) [24]. DQN [22] is a value-based method where the Q-value is represented by a neural network. The optimal policy is derived from the learned Q-value model. PPO [24] is a policy-gradient method where the policy is represented by a neural network and is directly sampled from the learned model. ACER [23] combines the idea of policy gradient and value-based methods.

We compare the performance of these three methods without the safety layer to exclude its effect on the learning algorithms. We perform a grid search for the hyperparameters for these three methods and select the best hyperparameters for each model. Table II shows that policies trained with PPO

TABLE II

RESULTS OF PRELIMINARY ALGORITHM COMPARISON

Parameter	PPO	ACER	DQN
Reached goal frequency	93.5 %	91.8 %	89.7 %
Elapsed training time	5.76 h	6.41 h	11.71 h
Multiprocessing	True	True	False

reached the goal most often on the test dataset and have the shortest computation time. Based on the implementation in OpenAI [25], PPO and ACER support multiprocessing to decrease the training time, while the DQN algorithm does not support parallelization. Therefore, we select PPO as our learning algorithm.

To be able to differentiate the PPO objective function for discrete action spaces, we apply the Gumbel noise from [26] to the output of the policy network:

$$a(t) = \underset{a_i(t)}{\arg\max}\,[\log(y_i(s_t)) \underbrace{-\log(-\log(u_i))}_{\text{Gumbel noise}}], \tag{4}$$

where $\log(y_i(s_t))$ is the output of the policy network corresponding to action a_i, and u_i is a random variable sampled from a uniform distribution $u_i \sim \mathcal{U}[0, 1]$. We use $\log(y_i(s_t))$ as the output of the policy network instead of $y_i(s_t)$ because we use a hyperbolic tangent as the activation function.

The optimization objective of PPO $J_{\text{PPO}}(\theta)$ [24] is

$$J_{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{\text{C}}(\theta) - v_{\text{l}} L_t^{\text{VF}}(\theta)\right], \text{with} \tag{5a}$$

$$L_t^{\text{C}}(\theta) = \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \tag{5b}$$

$$L_t^{\text{VF}}(\theta) = (V_\theta(s_t) - V_t^{\text{targ}})^2, \tag{5c}$$

where the probability ratio $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, $\pi_{\theta_{\text{old}}}$ is the old policy before the update, v_{l} and ϵ are scalar hyperparameters, V_θ is the estimated value function, and V_t^{targ} is the target value function collected through Monte Carlo simulations. The operator clip() limits the first argument to the range which is defined by the following two arguments, and $\hat{\mathbb{E}}_t[...]$ is the empirical mean over a finite batch of samples. We denote by θ the trainable parameters of the network. The main term of the objective is the clipped objective $L_t^{\text{C}}(\theta)$, which is easier to implement than using the Kullback Leibler divergence while showing stability comparable to trust-region-based methods. The value loss $L_t^{VF}(\theta)$ is necessary because parameters between policy network and value function network are shared. The advantage function $\hat{A}_t$ is estimated by a general advantage estimator [27].

The gradient of the PPO objective $J_{\text{PPO}}(\theta)$ is obtained by differentiation with respect to the trainable parameters θ, which is derived as in [27]:

$$\nabla_\theta J_{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t\left[\nabla_\theta \log\left(\pi_\theta(a_t|s_t)\right) J_{\text{PPO}}(\theta)\right], \tag{6}$$

where ∇_θ denotes the gradient with respect to the trainable parameters.

E. Action Masking

The safety layer generates a mask for the action to restrict the action space to the safe subspace. The safety mask $\mathbf{m}_{a_i}$ is a binary vector defined as

$$\mathbf{m}_{a_i}(t) = \begin{cases} 1, & \text{if } a_i(t) \text{ is verified safe} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

When we insert the safety mask (7) in (4), we obtain

$$\begin{aligned} a(t) &= \underset{a_i(t)}{\arg\max}\,[\log(y_i(s_t))\mathbf{m}_{a_i(t)} - \log(-\log(u_i\,\mathbf{m}_{a_i(t)}))] \\ &= \underset{a_i(t)\in\mathcal{A}_{\text{safe}}}{\arg\max}\left[\log(y_i(s_t)) - \log(-\log(u_i))\right], \end{aligned} \tag{8}$$

where $\mathcal{A}_{\text{safe}}$ is the set of actions which are verified as safe by the safety layer. The second line of (8) is derived from the fact that $0 - \log(-\log(0)) = -\infty$, i.e., unsafe actions can never be selected by the $\arg\max$ operator.

In the following, we show that masking does not affect the PPO objective and its gradient. All variables in (5) associated to masking are denoted by $\square^{\text{m}}$. First, we obtain the objective with masking $J_{\text{PPO}}^{\text{m}}(\theta)$ based on (5a):

$$J_{\text{PPO}}^{\text{m}}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{\text{C,m}}(\theta) - v_{\text{l}}\, L_t^{\text{VF}}(\theta)\right], \tag{9}$$

where $L_t^{\text{C,m}}(\theta)$ is the clipped objective, and $L_t^{\text{VF}}(\theta)$ is the value function loss (see (5c)). The masking of actions does not alter the value function loss term $L_t^{\text{VF}}(\theta)$ because the value function describes the value of a specific state independent of the actions. The clipped objective $L_t^{\text{C,m}}(\theta)$ is based on (5b) and is specified as

$$L_t^{\text{C,m}}(\theta) = \min(r_t^{\text{m}}(\theta)\hat{A}_t, \text{clip}(r_t^{\text{m}}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \tag{10}$$

where

$$r_t^{\text{m}}(\theta) = \frac{\pi_\theta^{\text{m}}(a_t, \mathbf{m}_{a_t}|s_t)}{\pi_{\theta_{\text{old}}}^{\text{m}}(a_t, \mathbf{m}_{a_t}|s_t)} = \frac{\pi_\theta^{\text{m}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}^{\text{m}}(a_t|s_t)}. \tag{11}$$

The policy values π_θ^{m} and $\pi_{\theta_{\text{old}}}^{\text{m}}$ for safe actions are not modified as the masking yields that a_t is always a safe action. Therefore, $r_t^{\text{m}}(\theta)$ stays the same as the initial (see r_t). The estimated advantage $\hat{A}_t$ depends on the value function and is not affected by the masking. Thus, the action masking does not affect the clipped objective $L_t^{\text{C,m}}(\theta)$ and the PPO objective $J_{\text{PPO}}^{\text{m}}(\theta)$. Consequently, the gradient $\nabla_\theta J_{\text{PPO}}^{\text{m}}(\theta)$ also remains the same as defined in (6).

IV. EXPERIMENTS

We demonstrate the proposed approach using a real-world highway dataset. To show the safety layer's effect on an RL agent, we train two groups of agents with and without the safety layer, respectively. Furthermore, we investigate the impact of different neural network structures. We evaluate the agents' performance by comparing their learning curves during training as well as the collision rate and goal-reaching rate on a test set.

A. Simulation Environment

a) Dataset: We utilize the highD dataset [28] to train and test the proposed approach. This dataset includes real traffic scenarios of German highways from six different locations with three-lane and two-lane roads. The dataset includes $5.1\,\mathrm{h}$ of recorded traffic with a time step size of $0.04\,\mathrm{s}$. The scenarios' duration ranges from $12.45\,\mathrm{s}$ to $12.67\,\mathrm{s}$ in the $95\,\%$ confidence interval. The scenarios' duration is similar because the observed road length is the same for the six locations, and the scenarios were generated from the original data. We generated tasks by removing a vehicle from the recorded data and using its start and the final state as the initial state and the center of the goal region, respectively. In particular, the goal area is the occupancy of the removed vehicle at its final position. The goal is reached if the ego vehicle intersects with the goal area. We randomly split the dataset into $80\,\%$ training set and $20\,\%$ test set.

b) Policy Network: We conducted experiments with two different types of policy networks, namely, a multi-layer perceptron (MLP) network [29] and a long short-term memory (LSTM) network [30]. The hyperparameters of the policy networks are determined using a grid search. The hyperparameter search compares the convergence and final rewards on the training set. The MLP network consists of three hidden layers, with 128 neurons in each layer. We choose an LSTM network as the second type because the task is time-sequential and recurrent networks are well suited to solve sequential tasks. However, training an MLP network is more stable as it converges for a larger variety of hyperparameters. The LSTM network consists of 128 neurons, and layer normalization is applied. Furthermore, we compute the running mean and standard deviation of the states to normalize the state space for both policy networks.

c) Training Mode: We conducted the training in two modes:

- Safe mode: we train the agent as proposed in Fig. 1.
- Non-safe mode: we exclude the safety layer.

In both modes, we restrict the action space to the corresponding high-level lane change action in case a lane change is currently conducted. Thereby, we ensure that a lane change cannot be prematurely aborted. A lane change is considered finished when the orientation of the ego vehicle differs at most by $0.2\,\mathrm{rad}$ from the orientation of the target lane, and the center of the ego vehicle is closer than one-fourth of the lane width to the centerline of the target lane.

B. Results

We trained the MLP and LSTM agents in safe and non-safe mode, resulting in four different agents. We compared these agents with respect to training performance, safety during training, and testing performance. Furthermore, we evaluated the effect of the safety layer on the agent using the test dataset.

a) Training Performance: Figure 4(a) shows the reward curves, which reveals that training in safe mode leads to faster convergence. Notably, training the agents in the non-safe mode required three million training steps; in contrast,

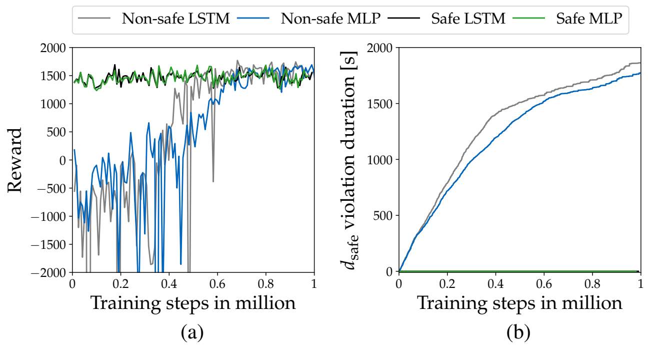

Fig. 4. Training results: (a) Reward curves for trained agents, (b) Safe distance violation for trained agents

one million steps were enough to train the agents in safe mode. The high negative rewards for non-safe mode agents originate from scenarios where the agent maintains a safe distance to the leading vehicle or changes lane close to the following vehicle on the target lane. The penalty for a safe distance violation with respect to surrounding vehicles is computed using (2e). Therefore, the penalty reaches high values if the agent almost collides with a traffic participant ahead.

Another indicator of how well the agents are exploring the action space is the number of lane changes per traffic scenario. Initially, the lane change frequency for training is about one lane change per traffic scenario for the safe agents and five for non-safe agents. The lower lane change frequency for the safe agents at the beginning of the training is due to the restriction of the action space. During training, the lane change frequency converges to about 0.2 lane changes per scenario, which means that the agent performs one lane change in every five scenarios with an average scenario duration of $12\,\mathrm{s}$. This convergence is significantly faster for the agents trained in safe mode. Note that the original data in the highD dataset has 0.1 lane changes per scenario, potentially caused by the low traffic density. Thus, lane change behaviors are not necessary in most scenarios.

Comparing the network types on the training set, the performance of MLP agents is almost identical to the corresponding LSTM agents. In particular, for the non-safe agents, the training converges marginally faster for the LSTM agent than for the MLP agent. For safe agents, there exists no visible difference in training convergence. For all agents, utilizing an MLP network leads to reaching the goal marginally more often.

b) Safety during Training: We have to differentiate between collisions for which the ego vehicle is responsible and collisions that occur because no interaction between traffic participants was considered due to prerecorded data. We exclude scenarios with collisions not caused by the ego vehicle from our evaluation, e.g., another vehicle colliding with the rear of the ego vehicle. Furthermore, the ego vehicle considers the safe distance to the leading vehicle and the following vehicle after a lane change.

During training, the non-safe agents caused collisions with

TABLE III

Final episode status on the test dataset

Agent	Collision	Reached goal
Non-safe LSTM	1.3 %	95.0 %
Non-safe MLP	0.8 %	**97.1 %**
Safe LSTM	**0.0 %**	87.5 %
Safe MLP	**0.0 %**	75.4 %

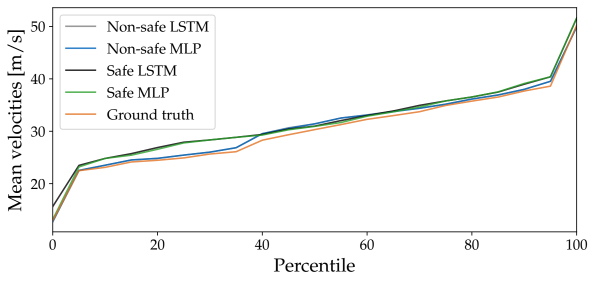

Fig. 5. Percentile curve for mean velocities on test scenarios

other traffic participants, while the safe agents did not cause any collisions. Moreover, the non-safe agents reached the goal more frequently than safe agents. However, the safe agents reach the goal in about 80 % of the scenarios in the last 500 000 training steps.

We measure safety using the duration for which the agents violated the safe distance. Figure 4(b) shows the duration of safe distance violations during training. The non-safe agents violate the safe distance while the safe agents never violate the safe distance.

c) Testing Performance: Table III summarizes the results of testing the agents. The test dataset's performance is similar to the performance of the training dataset, indicating that the agents are not overfitted. In the test set, the non-safe agents reach the goal more frequently than the safe agents. Moreover, the non-safe agents cause collisions in contrast to the safe agents. The performance of both policy networks is very similar in general. Based on the frequency of reaching the goal, the safe LSTM agent performs better than the safe MLP agent. This performance might be due to the ability of LSTM to store temporal features, e.g., acceleration of surrounding vehicles, providing additional information for planning a safe motion.

Furthermore, to show that the safe models do not drive too conservatively and impede the traffic flow, we evaluated the behavior of the agents against the original human driver trajectory from which the initial state and goal area of the task were generated. In general, the trained agents show a behavior similar to the original driver. Figure 5 depicts the percentile curve of the mean velocity. The mean velocities for the original human driver and all trained agents are almost identical. By comparing the safe distance violations to the leading vehicle, we observe that the original driver violated the safe distance, and the non-safe agents violated the safe distance even more. In contrast, safe agents did not violate any safe distances during testing.

d) Effect of Safety Layer on Learning: We tested the agents trained in safe mode also in non-safe mode to detect if training with the safety layer leads to better-performing agents. The comparison of the agents' performance in non-safe mode shows that the agents trained in safe mode perform worse than the agents trained in non-safe mode. The goal-reaching rate is 25 % less for the safe LSTM and 62 % less for the safe MLP agent than for agents trained in non-safe mode. Moreover, the agents trained in safe mode collided in more test scenarios (23 % for LSTM and 27 % for MLP). In contrast, the agents trained in non-safe mode caused collisions in about 1 % of the test scenarios. This performance is because the agents trained in safe mode did not experience dangerous situations with high penalties during training and cannot solve them in the non-safe test setting. Thus, the safety layer is necessary during deployment to ensure safety.

If training is conducted in a simulation setting and not in the real world, safety guarantees for real-world deployment would often suffice. Therefore, we tested the non-safe agents in safe mode and compared them to the agents trained and tested in safe mode. The performance of agents trained in non-safe mode and the safe LSTM agent on the test set is almost identical because all the agents reached the goal in 87 % of the scenarios. The safe MLP agent performs marginally worse as it only reaches the goal in 75 % of the scenarios. Due to the safety layer, none of the agents cause collisions in the test set. The result shows that the agents can be trained in non-safe mode and deployed in safe mode without causing performance loss and safety reduction. However, training in safe mode converges faster, which is a reason for training in safe mode.

C. Discussion

Although the proposed approach guarantees safety in all scenarios, the agent drives more cautiously than normal drivers, especially in dense traffic. In such scenarios, the predicted occupancy leads to a comparably small free space for the agent to drive. In this type of scenario, the interaction between traffic participants is essential. A traffic simulator can predict interactions between the agent and traffic participants to a certain degree.

Moreover, accurate modeling of physical parameters is crucial for set-based predictions. Too large physical bounds lead to significant over-approximation errors, limiting model applicability. Simultaneously, too small physical bounds cause inaccurate prediction that does not enclose the actual behavior, leading to unsafe behaviors. To check whether the modeled physical parameters over-approximate the real behavior, one can perform a reachset conformance test as shown for a pedestrian model in [31].

Due to the computational overhead for determining safe actions, the computation time for training safe agents is 16 times higher than for the non-safe agents. The average training step for safe agents takes 5.46 s and 0.112 s for non-safe agents. This significant increase in the training time is mainly because instead of one trajectory for the selected action, all possible trajectories are generated and compared to the predicted occupancies of traffic participants. Optimizing the current implementation is necessary to benefit from the faster convergence of safe agents in order to safeguard machine learning in real vehicles.

V. Conclusions

In this paper, we present a framework for safeguarding an RL agent using a safety layer to verify whether the proposed actions are safe and provide a provably safe fail-safe controller. Safe actions are determined by set-based prediction, which considers all possible motions of traffic participants. We evaluated the proposed approach using a real-world highway dataset. The result of the evaluation shows that the trained policy does not cause any collisions. Furthermore, the safe agent's ability to reach the goal region is comparable to that of non-safe agents. The proposed approach only requires an additional navigation system to realize basic motion planning on highways.

Acknowledgment

The authors gratefully acknowledge the partial financial support of this work by the German Research Foundation Grant AL 1185/3-2 and the research training group CONVEY funded by the German Research Foundation under grant GRK 2428.

References

[1] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8662–8668.

[2] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[3] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1469–1476.

[4] N. Fulton and A. Platzer, "Verifiably safe off-model reinforcement learning," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2019, pp. 413–430.

[5] S. Pathak, L. Pulina, and A. Tacchella, "Verification and repair of control policies for safe reinforcement learning," *Applied Intelligence*, vol. 48, no. 4, pp. 886–908, 2018.

[6] 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 *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2156–2162.

[7] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–6.

[8] G. R. Mason, R. C. Calinescu, D. Kudenko, and A. Banks, "Assured reinforcement learning for safety-critical applications," in *Doctoral Consortium at the 10th International Conference on Agents and Artificial Intelligence*, 2017.

[9] A. Akametalu, S. Kaynama, J. Fisac, M. Zeilinger, J. Gillula, and C. Tomlin, "Reachability-based safe learning with Gaussian processes," in *IEEE Conference on Decision and Control*, 2015, pp. 1424–1431.

[10] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[11] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015.

[12] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 353–359.

[13] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[14] T. Fraichard and H. Asama, "Inevitable collision states a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[15] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[16] M. Althoff, S. Maierhofer, and C. Pek, "Provably-correct and comfortable adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, 2020.

[17] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 485–491.

[18] M. Althoff, "CommonRoad: Vehicle models." [Online]. Available: https://commonroad.in.tum.de/

[19] C. Pek and M. Althoff, "Efficient computation of invariably safe states for motion planning of self-driving vehicles," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 3523 – 3530.

[20] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1686–1693.

[21] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 987–993.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *Twenty-seventh Conference on Neural Information Processing Systems – Workshop on Deep Learning*, 2013.

[23] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *International Conference on Learning Representations (ICLR)*, 2017.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "OpenAI baselines," 2017. [Online]. Available: https://github.com/openai/baselines

[26] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-softmax," in *International Conference on Learning Representations (ICLR)*, 2016.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *4th International Conference on Learning Representations, ICLR*, 2016.

[28] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125.

[29] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2007.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts and London, England: MIT Press, 2016.

[31] S. B. Liu, H. Roehm, C. Heinzemann, I. Lütkebohle, J. Oehlerking, and M. Althoff, "Provably safe motion of mobile robots in human environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep 2017.