Falsification-Based Robust Adversarial Reinforcement Learning

Xiao Wang[1], Saasha Nair[1], and Matthias Althoff[1]

***Abstract*— Reinforcement learning (RL) has achieved enormous progress in solving various sequential decision-making problems, such as control tasks in robotics. Since policies are overfitted to training environments, RL methods have often failed to be generalized to safety-critical test scenarios. Robust adversarial RL (RARL) was previously proposed to train an adversarial network that applies disturbances to a system, which improves the robustness in test scenarios. However, an issue of neural network-based adversaries is that integrating system requirements without handcrafting sophisticated reward signals are difficult. Safety falsification methods allow one to find a set of initial conditions and an input sequence, such that the system violates a given property formulated in temporal logic. In this paper, we propose falsification-based RARL (FRARL): this is the first generic framework for integrating temporal logic falsification in adversarial learning to improve policy robustness. By applying our falsification method, we do not need to construct an extra reward function for the adversary. Moreover, we evaluate our approach on a braking assistance system and an adaptive cruise control system of autonomous vehicles. Our experimental results demonstrate that policies trained with a falsification-based adversary generalize better and show less violation of the safety specification in test scenarios than those trained without an adversary or with an adversarial network.**

I. Introduction

Recent advancements, such as superhuman performance in a range of Atari games in 2015 [1], followed by AlphaGo's victory against the human world champion in Go in 2016 [2], [3], have developed numerous research interests in reinforcement learning (RL) [4]. Consequently, RL has greatly progressed in real-world applications, such as robotics [5], natural language processing [6], and autonomous driving [7], [8]. However, RL still suffers from some shortcomings, such as bad generalization in real-world scenarios, risk-sensitive reward functions, and violation of safety constraints [9], [10]. This study addresses the generalization problem due to the huge amount of training required for RL.

Pinto et al. [11] discussed generalization by adding disturbances as adversarial examples [12]. Their study was later extended by Pan et al. [13]. By training with adversarial examples, the authors reduced the simulation-to-reality gap caused by modeling errors, so the trained models generalize better in real-world scenarios. Adversarial RL is formulated as a two-player zero-sum game in [11], [13], in which an adversary aims to obstruct the success of the learning system. However, learning in a zero-sum game requires finding a Nash equilibrium, which is especially challenging for continuous high-dimensional problems [14]. Otherwise, if we formulate the problem as a non-zero-sum game, in which the adversary optimizes a different reward function, then a sophisticated reward function for the adversary would have to be handcrafted. One can argue that engineering the reward function could improve the generalization ability of an RL agent. However, as pointed out in [10], designing a *perfect* reward function is a very challenging task. For instance, the traffic rule *a vehicle is not allowed to overtake another on its right side except in congested traffic* requires a sequence of events, which is difficult to integrate into a reward function; the traffic rule can easily be expressed by a temporal logic. Therefore, temporal logic falsification methods provide a possibility to automatically improve generalization without having to tune the reward functions.

In this paper, we propose a new framework: we develop adversarial samples in a single RL agent setting, wherein the protagonist is represented by an RL agent, while safety falsification methods act as an adversary. Safety falsification approaches drive a system to unsafe behaviors, which violate given safety specifications [15].

The remainder of this paper is organized as follows. Section II provides an overview of current solutions in related studies for adversarial RL and system falsification for safety-critical systems. Section III introduces falsification-based RARL, which is evaluated in Section IV. In Section V, we present the conclusions and potential future research directions.

II. Related Work

A. Adversarial Reinforcement Learning

Despite the success of RL algorithms, they are susceptible to changes in environmental settings [10], [16]. Hence, various forms of adversarial training [12] have been introduced to solve this problem. One such approach involves adding adversarial perturbations to the observations of the agent by attacking either only the image inputs [17]–[19] or addressing the entire state vector [20], [21]. Another approach involves using source-domain ensembles, which are adapted to the target domain using the Bayesian model adaptation [22].

Further, the approach most relevant to our study is the minimax approach extending robust RL [16]. This approach, which is known as robust adversarial RL (RARL) [11], simultaneously trains two RL agents: one called the protagonist, while the other is called the adversary. The adversary is tasked with applying destabilizing forces to impede the protagonist, while the protagonist learns to be robust to the adversary. An extension of RARL has been provided

[1]The authors are with the Department of Informatics, Technische Universität München, 85748 Garching, Germany.
xiao.wang@tum.de, saasha.nair@tum.de, althoff@in.tum.de

by risk-averse RARL (RARARL) [13], which focuses on safety-critical cyber-physical systems, by modeling the risk as the variance of an ensemble of value functions. However, RARARL can only solve problems with a discrete action space and requires an ensemble of multiple neural networks, requiring a significant computational resource. In contrast to the RARL and RARARL that model the setup as a two-agent RL scenario, as mentioned in Section I, our proposed solution reduces the number of reward functions to be defined and tuned, requires fewer parameters of the adversary to be optimized, as introduced in Section III-A, and allows for better expressiveness for the adversary using temporal logic specifications.

B. Safety Falsification

Safety falsification methods aim at finding initial conditions and input sequences, with which a system violates a given safety specification. Two categories of various approaches exist for solving this problem. We first review *single-shooting* methods, which simulate trajectories from specific initial conditions and input traces and iterate until a falsifying trajectory is obtained. A single-shooting method is achieved by applying Monte-Carlo methods [15], [23], [24], ant colony optimization method [25], cross-entropy method [26], or rapidly-exploring random tree search [27]–[30]. A *multiple-shooting* approach is proposed in [31], [32] to split system trajectories into small segments by simulating from multiple initial conditions in a state space decomposed into cells. Once a segment reaches an unsafe state, the cell size is refined until the segments can be concatenated to a complete system trajectory. The ant-colony method and the Monte-Carlo method were compared in [25] for two benchmarks, and similar results were obtained. As shown in [26], the cross-entropy method outperformed the Monte-Carlo method on five benchmarks. Hence, we employ the cross-entropy method in this study. Note that our framework can also use other falsification approach.

III. FALSIFICATION-BASED RARL

A. Safety Falsification

To formulate the safety falsification problem, we first introduce several important definitions adopted from [15]. A dynamic system Σ can be regarded as a mapping from initial states $\boldsymbol{x}_0 \in \mathcal{X}_0 \subset \mathbb{R}^n$ and input signals $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$ to output signals $\boldsymbol{y} \in \mathcal{Y} \subset \mathbb{R}^k$: $\mathcal{X}_0 \times \mathcal{U} \to \mathcal{Y}$.

We formulate system properties in metric temporal logic (MTL) [33]. A temporal logic combines propositions of classical logics with time dependence such that a truth value is assigned to each atomic proposition at each time instant [34]. An atomic proposition p is a statement that can be either *true* or *false*. Atomic propositions and the logical connectives, such as Boolean operators *not* and *or* denoted by $\neg$ and $\vee$, form propositional formulas. The temporal operator *until*, which is denoted by $\boldsymbol{U}$, indicates that in a formula $\varphi_1 \boldsymbol{U} \varphi_2$, the first formula φ_1 holds *until* the second formula φ_2 holds; the time t when φ_2 starts to hold is unconstrained, i.e., $t \in (0, \infty)$. The temporal operator *globally*, which is denoted by $\boldsymbol{G}$, indicates that formula φ must hold for all times. In addition, MTL is an extension of temporal logic in which temporal operators are replaced by time-constrained operators. Thus, $\boldsymbol{U}$ is replaced by $\boldsymbol{U_I}$, where $\boldsymbol{I} \subseteq (0, \infty)$, indicating that t is constrained by $\boldsymbol{I}$. The syntax of an MTL formula φ is defined as follows [24]:

$$\varphi := true \,|\, p \,|\, \neg\varphi \,|\, \varphi_1 \vee \varphi_2 \,|\, \varphi_1 \boldsymbol{U_I} \varphi_2 | \boldsymbol{G}\varphi, \tag{1}$$

which indicates that the value of an MTL formula is always *true* or *false*. Other logical expressions can be formed from logical equivalences, such as $a \implies b \equiv \neg a \vee b$. In this study, we use the *global* operator $\boldsymbol{G}$ as presented in (9). More general formulas can be obtained from (1).

Definition. (MTL Falsification). For an MTL specification φ, the MTL falsification problem aims to find initial states $\boldsymbol{x}_0 \in \mathcal{X}_0$ and an input sequence $\boldsymbol{u} : [0, T] \to \mathcal{U}$ such that the resulting trajectory $\boldsymbol{y}$ of system Σ violates the specification φ, which is denoted by

$$\boldsymbol{y}(\boldsymbol{x}_0, \boldsymbol{u}) \not\models \varphi. \tag{2}$$

Naïve falsification uniformly samples the set of initial conditions and input sequences. A more efficient approach is to guide the search using a metric, measuring the distance between the trajectory and set of states violating the specification. A *robustness metric* ε is proposed in [35] to express the satisfaction of an MTL property over a given trajectory as a real number instead of a Boolean value (0 for no intersection with unsafe sets and 1 for successful falsification). The sign of ε reveals whether a trajectory $\boldsymbol{y}$ satisfies an MTL property φ. The robustness of $\boldsymbol{y}$ with respect to φ is denoted by

$$\varepsilon = [\![\varphi]\!]_d(\boldsymbol{y}, t), \tag{3}$$

and defined as follows [24]–[26]:

$$\begin{aligned}
[\![true]\!]_d(\boldsymbol{y}, t) &:= +\infty \\
[\![p]\!]_d(\boldsymbol{y}, t) &:= \mathbf{Dist}_d(\boldsymbol{y}(t), \mathcal{O}(p)) \\
[\![\neg\varphi]\!]_d(\boldsymbol{y}, t) &:= -[\![\varphi]\!]_d(\boldsymbol{y}, t) \\
[\![\varphi_1 \vee \varphi_2]\!]_d(\boldsymbol{y}, t) &:= \max([\![\varphi_1]\!]_d(\boldsymbol{y}, t), [\![\varphi_2]\!]_d(\boldsymbol{y}, t)) \\
[\![\varphi_1 \boldsymbol{U_I} \varphi_2]\!]_d(\boldsymbol{y}, t) &:= \sup_{t' \in (t + \boldsymbol{I})} \min([\![\varphi_2]\!]_d(\boldsymbol{y}, t'), \\
&\qquad \inf_{t < t'' < t'} [\![\varphi_1]\!]_d(\boldsymbol{y}, t'')),
\end{aligned} \tag{4}$$

where $\mathcal{O}(p)$ denotes the set in which p is fulfilled. The signed distance denoted by $\mathbf{Dist}_d$ is defined as

$$\mathbf{Dist}_d(\boldsymbol{y}, \mathcal{O}) := \begin{cases} -\inf\{d(y, x) \,|\, x \in \mathcal{O}\} & \text{if } y \notin \mathcal{O} \\ \inf\{d(y, x) \,|\, x \in \mathcal{X} \setminus \mathcal{O}\} & \text{if } y \in \mathcal{O}, \end{cases} \tag{5}$$

where $d(y, x)$ is typically defined as the Euclidean distance for continuous systems:

$$d(y, x) = ||y - x||^2. \tag{6}$$

Consequently, (2) can be defined as a minimization problem:

$$\min_{\boldsymbol{x}_0 \in \mathcal{X}_0, \boldsymbol{u}:[0,T] \to \mathcal{U}} [\![\varphi]\!]_d(\boldsymbol{y}(\boldsymbol{x}_0, \boldsymbol{u}), t). \tag{7}$$

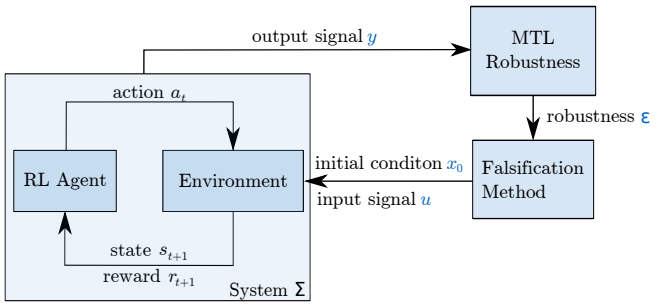


Fig. 1. Falsification-based RARL framework. The RL agent and environment are regarded as the black box system Σ. Falsification model serves as the adversary, which provides input sets x_0 and u for the environment such that the system trajectory falsifies MTL specifications. The RL agent is trained further under the falsified environment.

The cross-entropy method combines piecewise-uniform and Gaussian distributions to approximate the underlying distribution of the robustness value in (3) over the set $\mathcal{X}_0 \times \mathcal{U}$ [26]. The proposed distribution is denoted by p_θ with parameter θ, while the unknown real distribution is denoted by q. The distance between the two distributions is measured using the Kullback-Leibler divergence [36]:

$$D(q,\, p_\theta) = \int_{\mathcal{X}_0 \times \mathcal{U}} \log\left(\frac{q(\xi)}{p_\theta(\xi)}\right) q(\xi)\, \mathrm{d}\xi, \tag{8}$$

where $\xi \in \mathcal{X}_0 \times \mathcal{U}$. Since the actual distribution q is unknown, $D(q,\, p_\theta)$ is estimated using N_s sampled data points, which are chosen by the current approximation p_θ. The samples are sorted through their robustness values, and the m least robust samples are considered, with $m \ll N_s$. Then, parameter θ is updated by minimizing the divergence $D(q,\, p_\theta)$ over m data samples. Moreover, this procedure iterates until the divergence converges to a threshold. Afterward, the initial conditions and input sequences are sampled based on the converged distribution p_θ.

In this study, we consider an autonomous vehicle on a highway with two safety requirements: the agent is not allowed to collide with the leading vehicle or to drive backward on the highway at all times. Therefore, we formulate these requirements as follows:

$$\boldsymbol{G}(\neg\varphi_{\text{collision}} \wedge \neg\varphi_{\text{reverse}}). \tag{9}$$

Note that the proposed method can be directly applied to more complicated specifications containing temporal operators, such as *until* and *eventually*. In the future, we will integrate more traffic rules in the system requirements as proposed in [37]–[41].

B. Falsification-Based RARL

Figure 1 depicts our framework of falsification-based RARL, and Algorithm 1 presents our approach in detail. We formulate our RL problem as a Markov decision process (MDP) defined by a 5-tuple (S, A, P, R, γ), where S denotes the state space, A denotes the action space, P denotes the state transition probability, R denotes the expected reward signal, and $\gamma \in [0, 1]$ denotes the discount factor. Further, the left part of Fig. 1 describes the learning process of the RL policy. The agent acts on the environment, followed by updating its state and reward. Our experiments reveal that the policy converges slower if the adversary interferes too early, as was also observed in [13]. Therefore, as shown in Algorithm 1 line 2-8, we first train the agent for t_f time steps without an adversary (see Fig. 3 and Fig. 4). Next, we regard our trained model and the environment as a black box system Σ. As illustrated in the right part of Fig. 1, we employ the cross-entropy method to obtain initial conditions and input sequences for the environment (the behavior of other vehicles) under which the agent violates our MTL specification (9). We initialize our environment with the initial conditions, change the behavior based on the new input sequences, and train the policy further in the new adversarial environment (line 13). This procedure is repeated until the policy converges to zero violations.

Algorithm 1: Falsification-Based RARL

Input: Training steps T; environment E; number of actors n_a; time steps each actor runs at each iteration t_a; MTL specification φ; time step to start falsification t_f; number of falsification iterations n_f
Initialize: Parameters of policy and value network ϕ_0
Result: Trained policy and value network ϕ

1 **while** $t < T$ **do**
2 **for** actor $= 1, 2, ..., n_a$ **do**
3 Run policy ϕ_{old} in E for t_a time steps ;
4 Compute advantage estimates $\hat{A}_1, ..., \hat{A}_{t_a}$ (10);
5 **end**
6 Optimize surrogate $L^{\text{CLIP+VF}}$ (11) wrt. ϕ with batch size $n_a t_a$;
7 $\phi_{\text{old}} \leftarrow \phi$;
8 $t = t + n_a t_a$;
9 **if** $t > t_f$ **then**
10 Initialize falsifier parameter θ_0 ;
11 **for** iter $= 1, 2, ..., n_f$ **do**
12 Sample input conditions $\boldsymbol{x}_0, \boldsymbol{u}$ from $p_{\theta_{\text{old}}}$;
13 Initialize new environment E_{iter} with $\boldsymbol{x}_0$, change the behavior of E_{iter} with $\boldsymbol{u}$;
14 Collect trajectories $\boldsymbol{y}$ in E_{iter} with agent ϕ_{old} and evaluate robustness value according to (3) ;
15 Estimate (8) with $\boldsymbol{y}$ and minimize (8) wrt. θ ;
16 $\theta_{\text{old}} \leftarrow \theta$;
17 **end**
18 $E \leftarrow E(\boldsymbol{x}_0, \boldsymbol{u})$;
19 **end**
20 **end**

We choose proximal policy optimization (PPO) [42] to optimize the policy network due to its superior performance in continuous control problems when compared with other

state-of-the-art approaches. To reduce variance, we use an actor-critic architecture [43] to approximate both the policy and the value function with neural networks. Additionally, we estimate the advantage function $\hat{A}_t$ using a general advantage estimator (GAE) [44] as follows:

$$\begin{aligned} &\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \\ &\text{with} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \end{aligned} \tag{10}$$

where $\lambda \in [0, 1]$ denotes a discount factor of the advantage estimator and makes a compromise between variance and bias. The objective function of PPO is defined as follows:

$$L^{\text{CLIP+VF}}(\phi) = \hat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\phi) - cL_t^{\text{VF}}(\phi) \right], \text{with} \tag{11a}$$

$$L^{\text{CLIP}}(\phi) = \hat{\mathbb{E}}_t \left[\min(r_t(\phi)\hat{A}_t, \text{clip}(r_t(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{11b}$$

$$L_t^{\text{VF}}(\phi) = (V_\phi(s_t) - V_t^{\text{targ}})^2, \tag{11c}$$

where ϕ denotes the parameters of the policy and value network, the probability ratio is defined by $r_t(\phi) = \dfrac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{old}}(a_t|s_t)}$, $\pi_{\phi_{old}}$ denotes the old policy before the update, c and ϵ are hyper-parameters, V_ϕ denotes the estimated value function, V_t^{targ} is the target value function collected through Monte-Carlo simulations, clip is an operator for limiting the operand in a given range, and $\hat{\mathbb{E}}_t[...]$ denotes the empirical average over a finite batch of samples.

IV. EXPERIMENTS

We evaluate our approach on two systems. The first one is a braking assistance (BA) system of an autonomous vehicle applied to avoid rear-end collisions and driving reversely on a highway, while the second one is an adaptive cruise control (ACC) system that keeps a safe distance from the leading vehicle and follows the desired velocity. The two systems are implemented in the same traffic simulator with different reward functions, which are described in Section IV-B. To fairly evaluate the performance of our approach, we train each system in three environments: a baseline environment without an adversarial model, an adversarial environment with an RL agent as an adversary, and an adversarial environment with an adversary using our falsification method. The adversarial RL agent in the second environment is trained by utilizing RARL [11]. Moreover, because RARARL [13] was proposed to solve discrete action space problem and cannot be directly applied to problems with a continuous action space, we choose RARL over the more recent RARARL to train the adversarial RL agent. Following [11], we call the policy that controls the ego vehicle *protagonist*.

A. Dataset

In the adversarial environments, the behavior of the leading vehicle is altered by applying the falsification method or an adversarial RL agent. The baseline environment could be achieved by utilizing either rule-based driver models, such as the intelligent driver model (IDM) [45], or real traffic data. A key limitation of rule-based driver models is their homogeneity. The policy could easily overfit to react only to a particular behavior such that it fails to generalize, while driving behaviors from real traffic are more diverse. Therefore, we choose the recently published highway drone (HighD) dataset of naturalistic vehicle trajectories on German highways [46].

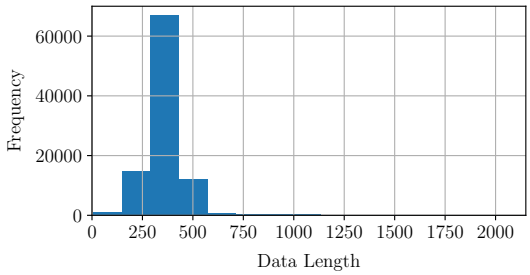


Fig. 2. Frequency histogram of the length of lane-following trajectories in HighD dataset [46].

HighD recorded $16.5\,\text{h}$ of video at six locations using a drone and it extracted over $45\,000\,\text{km}$ of vehicle trajectories at $25\,\text{Hz}$ using computer vision algorithms. Since the longitudinal driving behavior of a lane-changing vehicle differs from that of a lane-following vehicle, we filter out the trajectories of all lane-changing vehicles so that $97\,184$ lane-following trajectories remain. As depicted in Fig. 2, the frequency histogram demonstrated the distribution of the total length of these trajectories. To avoid overfitting, each trajectory should be used at most once during training. In addition, the original traffic scenarios cover a lane of $420\,\text{m}$ with a median duration of $13.6\,\text{s}$ for each vehicle. In this setting, the ego vehicle is less likely to encounter a critical situation. Therefore, we extend the lane length to $600\,\text{m}$ and the total time of a scenario to $20\,\text{s}$, i.e., 500 time steps. Therefore, to obtain sufficient trajectories, we select the longitudinal acceleration signals of lane-following trajectories with total time step $L \geq 250$, cut signals to 250 time steps, and append the signals with reversely duplicated signals. In all, we obtain $93\,454$ trajectories and separate them into $70\,\%$ and $30\,\%$ trajectories for training and testing, respectively.

B. Environment

We set up a driving simulator based on the CommonRoad benchmark suite [47] and OpenAI Gym [48]. Since the goal of the agent is to learn the longitudinal driving behavior, our simulator contains only a straight lane of $600\,\text{m}$. An episode terminates if the leading vehicle reaches the end of the lane, the maximal time step 500 is reached, a collision happens, or the ego vehicle drives in reverse. Both vehicles are driven based on a point-mass model, whose input is acceleration, which is sampled from the policy network. To ensure that the scenario is solvable, the leading vehicle is initially at least as far away from the ego vehicle as the safe distance. We assume that both vehicles have the same maximum deceleration $a_{\max} = 10\,\text{m/s}^2$. Then, the safe

distance is computed based on [49] as follows:

$$s_{\text{safe}} = \frac{1}{2a_{\max}}(v_f^2 - v_l^2) + v_f\delta, \tag{12}$$

where v_f and v_l denote the velocities of the following and leading vehicles, respectively, and δ is the reaction delay of the following vehicle. As assumed by [49], we utilize $\delta = 0.3\,\text{s}$ for autonomous vehicles.

Without loss of generality, the initial position of the ego vehicle is fixed at $s_{\text{ego}} = 10\,\text{m}$, whereas the initial position of the leading vehicle is either randomly sampled within the range $[s_{\text{ego}} + s_{\text{safe}},\ s_{\text{ego}} + s_{\text{safe}} + 40]$ or calculated by applying the falsification tool. The acceleration and initial velocity of the leading vehicle are either extracted from the selected HighD trajectories, or computed by utilizing the adversaries, e.g., in (7), $\boldsymbol{x}_0$ corresponds to the initial position and velocity of the leading vehicle, and $\boldsymbol{u}$ corresponds to the acceleration of the leading vehicle.

Since maintaining a safe distance from the leading vehicle is crucial for avoiding collision, the feature vector of the policy networks should provide all the necessary information to calculate the safe distance in (12). Thus, we choose the feature vector for both systems as presented in Tab. I.

The reward function of the protagonist of the BA system is defined as follows:

$$r_{\text{BA}} = \begin{cases} -1, & \text{if agent drives reversely or collision happens} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The reward function of the protagonist of the ACC system is defined as follows:

$$r_{\text{ACC}} = \begin{cases} -1, \text{if agent drives reversely or collision happens} \\ -0.1\exp\left(\frac{-5\,s}{s_{\text{safe}}}\right), \text{if } s < s_{\text{safe}} \\ -0.05\exp\left(\frac{-5\,v_{\text{ego}}}{v_{\text{leading}}}\right), \text{if } v_{\text{ego}} < v_{\text{leading}} \\ 0, \text{otherwise.} \end{cases} \tag{14}$$

The first and last items are the same as in r_{BA}. Additionally, two terms are added: the second term penalizes a violation of the safe distance using a nonlinear function that increases the penalization as the ego vehicle gets closer to the leading vehicle; the third term penalizes the ego vehicle for driving slower than the leading vehicle if the distance is greater than the safe distance. Note that the coefficients in (13) and (14) are selected through a grid search. The nonlinear functions in the second and third terms in (14) significantly increase the performance of the agent.

The goal of the adversarial policy is to minimize the reward of the protagonist. Therefore, we choose $r_{\text{adv}} = -r_{\text{BA}}$ and $r_{\text{adv}} = -r_{\text{ACC}}$ as the reward functions of the adversarial policies for the BA and ACC systems.

C. Baseline Model

As mentioned in Section III-B, we train our policies and value functions by utilizing PPO [42] and an actor-critic algorithm [43]. In particular, we use a shared network design to share the features between the policy and the value functions. Moreover, we build our models based on the implementation of OpenAI Baselines [50]. Since our goal is to compare all methods with the same hyper-parameters, we did not perform a hyper-parameter optimization for each method. We instead utilized the default hyper-parameters that the OpenAI Baselines [50] provides. The shared policy and value network has two hidden layers with 64 neurons each and *tanh* as its activation function. The model is optimized using Adam optimizer [51] with a learning rate of 0.0003 and a batch size of 128. In (10), the discount factor of the advantage estimator is $\lambda = 0.95$.

TABLE I
FEATURES USED BY POLICY NETWORK

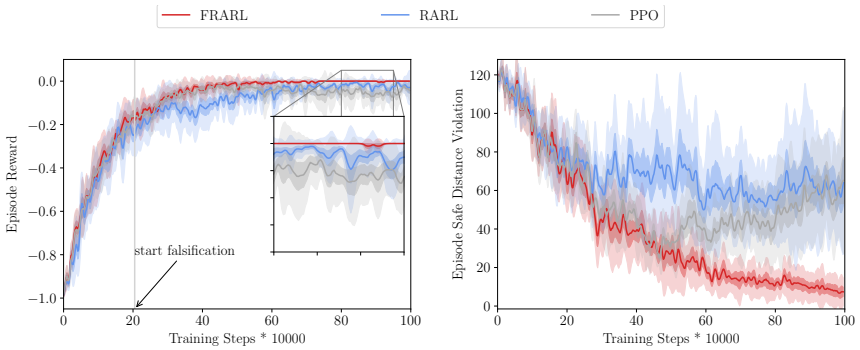
Feature	Units	Description
s	m	distance to leading vehicle
$v_{\text{ego}} - v_{\text{leading}}$	m/s	relative velocity to leading vehicle
v_{ego}	m/s	velocity of ego vehicle
a_{leading}	m/s^2	acceleration of leading vehicle
a_{ego}	m/s^2	acceleration of ego vehicle

In all conducted experiments, we first train the protagonist policy without the adversary for 200,000 training steps to allow it to learn basic skills as proposed in [13]. In the RL adversarial environment, we update the parameters of the protagonist θ_μ to maximize r_{BA} or r_{ACC} for $N_\mu = 10$ iterations, while the parameters of the adversary θ_ν are kept constant. Then, to maximize r_{adv}, we keep θ_μ constant and update θ_ν for $N_\nu = 1$ iteration. Here, N_μ and N_ν are empirically chosen. This process iterates until both policies converge. In the falsification adversarial environment, we apply S-Taliro [52], which is a MATLAB toolbox for MTL falsification for hybrid systems, to falsify the protagonist during training. S-Taliro is called every 10 iterations to compute 10 acceleration traces as well as initial positions and velocities for the leading vehicle, in which the protagonist falsifies the given specification (9). To train the policy further, the computed traces were randomly picked by the simulator. For the rest of this paper, we call the baseline model PPO, the policy model trained with an RL adversarial agent RARL, and the policy model trained with our method FRARL.

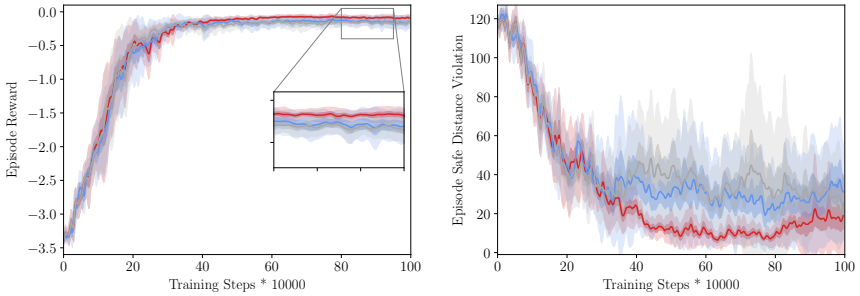
D. Evaluation

During the training phase, policies are set to be stochastic to encourage exploration, whereas during the evaluation and falsification phases, deterministic policies are used. To fairly compare the robustness of the three models, 10 policies with different random seeds are trained for each method. We evaluate the learning progress of all models in two groups of test scenarios, namely the HighD and random test scenarios, where the acceleration of the leading vehicle is randomly sampled in a given range. Note that we used random test scenarios instead of the adversarial or falsified scenarios because the agent is destined to encounter low reward in the adversarial and falsified scenarios.

We regard a model as *robust* if it satisfies the safety specification (9) in unseen scenarios, i.e., the HighD and

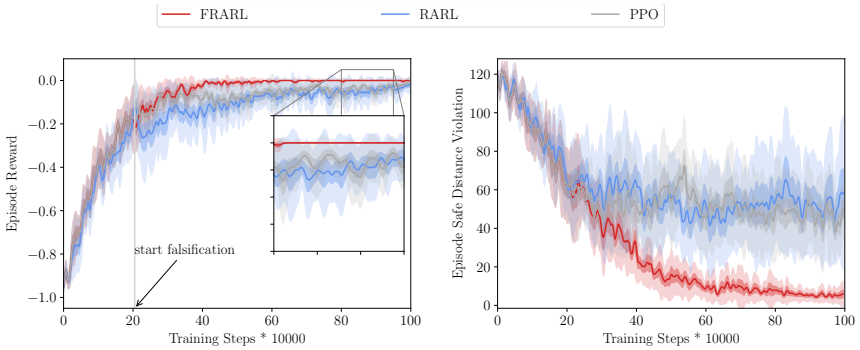


(a) Episode reward and safe distance violation curves on the HighD testing scenarios of the braking assistant system.

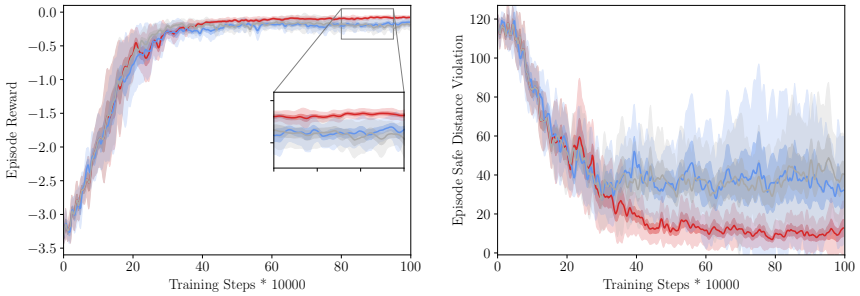


(b) Episode reward and safe distance violation curves on the HighD testing scenarios of the adaptive cruise control system.

Fig. 3. The learning curves of episode reward and number of safe distance violations on the **HighD** test scenarios of the BA and ACC system trained using PPO, RARL, and FRARL. For both systems, FRARL showed slightly higher episode reward and much less safe distance violations. In addition, for the BA system, FRARL converged to a zero reward at half of the training steps. Furthermore, FRARL had a lower variance for the episode reward and safe distance violations.



(a) Episode reward and safe distance violation curves on random testing scenarios of the braking assistant system.



(b) Episode reward and safe distance violation curves on random testing scenarios of the adaptive cruise control system.

Fig. 4. The learning curves of episode reward and number of safe distance violations on the **random** test scenarios of the BA and ACC system trained using PPO, RARL, and FRARL. FRARL showed more advantage on random than on HighD scenarios, indicating that training in a falsified environment improves the ability of an RL agent to generalize to unknown scenarios.

TABLE II

AVERAGE RATE OF UNSAFE BEHAVIORS OVER 28 037 HIGHD TEST SCENARIOS

	BA		ACC	
Violation	Reverse	Collision	Reverse	Collision
PPO	0.34%	4.59%	0.005%	0.24%
RARL	0.009%	2.70%	**0**	0.17%
FRARL	**0**	**0.015%**	**0**	**0**

TABLE III

AVERAGE RATE OF UNSAFE BEHAVIORS OVER 28 037 RANDOM TEST SCENARIOS

	BA		ACC	
Violation	Reverse	Collision	Reverse	Collision
PPO	0.40%	6.05%	0.028%	0.33%
RARL	0.026%	3.59%	0.013%	0.26%
FRARL	**0.0018%**	**0.025%**	**0**	**0**

random test scenarios. To analyze the safety of the behavior of the agent, we count the number of time steps in which the agent violates the safe distance to the leading vehicle. Figures 3 and 4 depict the episode reward and number of safe distance violations of the BA and ACC systems trained using PPO, RARL, and FRARL in the random and HighD test scenarios, respectively. For both systems, FRARL showed slightly higher episode reward and much less safe distance violations. In addition, for the BA system, FRARL converged to a zero reward at half of the training steps. Furthermore, FRARL had a lower variance for the episode reward and safe distance violations. It also showed more advantage on random than on HighD test scenarios, indicating that training in a falsified environment improves the ability of an RL agent to generalize to unknown scenarios.

To further address the robustness of the trained models, we evaluate all models on 28 037 HighD and random test scenarios and show the average rate of reverse driving and collisions in Tabs. II and III, respectively. For both systems on both test scenarios, FRARL achieved the lowest rate of collisions and reverse driving. FRARL outperformed RARL because an adversarial RL agent seeks scenarios in which the reward of the policy stays low but not necessarily safety-critical scenarios. Thus, after a policy converges to a good behavior, a further increase in safety becomes much more difficult for RARL. Our falsification method instead optimizes the scenarios until safety-critical scenarios are obtained. Therefore, the policies trained in safety-critical scenarios behave much safer than those classically trained.

V. CONCLUSIONS

We presented a framework for combining RL with safety falsification methods, which served as an adversarial RL, to improve the robustness of trained policies. By formulating safety requirements in metric temporal logics (MTLs), we spared ourselves the trouble of handcrafting a reward function for the adversary. For BA and ACC systems, we demonstrated that the policies trained using our approach satisfy the safety specification much better in test scenarios and thus are more robust. In the future, we will extend our experiments to more complex driving scenarios, such as urban scenarios. Further, we will integrate traffic rules in our MTL specifications.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support of this work by the German Research Foundation (DFG) under grant number AL 1185/3-2.

REFERENCES

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

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] T. Chouard, "The Go Files: AI computer wraps up 4-1 victory against human champion," *Nature News*, 2016.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[5] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[6] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy, "Deep reinforcement learning for sequence-to-sequence models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2469–2489, 2020.

[7] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8248–8254.

[8] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[9] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[10] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

[11] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proc. of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 2817–2826.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR*, 2014.

[13] X. Pan, D. Seita, Y. Gao, and J. Canny, "Risk averse robust adversarial reinforcement learning," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8522–8528.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[15] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, pp. 1–30, 2013.

[16] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[17] S. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *5th International Conference on Learning Representations, ICLR*, 2017.

[18] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *Proc. of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3756–3762.

[19] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," in *5th International Conference on Learning Representations, ICLR*, 2017.

[20] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3932–3939.

[21] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 2040–2042.

[22] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," in *5th International Conference on Learning Representations, ICLR*, 2017.

[23] H. Abbas and G. Fainekos, "Linear hybrid system falsification through local search," in *International Symposium on Automated Technology for Verification and Analysis*, 2011, pp. 503–510.

[24] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proc. of the 13th international conference on Hybrid systems: computation and control*. ACM, 2010, pp. 211–220.

[25] Y. S. R. Annapureddy and G. E. Fainekos, "Ant colonies for temporal logic falsification of hybrid systems," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2010, pp. 91–96.

[26] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *Proc. of the 15th international conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 125–134.

[27] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, 2004, pp. 142–156.

[28] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods Symposium*, 2015, pp. 127–142.

[29] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," in *NASA Formal Methods Symposium*, 2017, pp. 357–372.

[30] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2019.

[31] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski, "A trajectory splicing approach to concretizing counterexamples for hybrid systems," in *52nd Conference on Decision and Control*. IEEE, 2013, pp. 3918–3925.

[32] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski, "Multiple shooting, CEGAR-based falsification for hybrid systems," in *Proc. of the 14th International Conference on Embedded Software*. ACM, 2014, pp. 1–10.

[33] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[34] E. A. Emerson, "Temporal and modal logic," in *Formal Models and Semantics*, 1990, pp. 995–1072.

[35] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[36] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[37] A. Rizaldi and M. Althoff, "Formalising traffic rules for accountability of autonomous vehicles," in *Proc. of the IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 1658–1665.

[38] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow, "Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL," in *International Conference on Integrated Formal Methods*, 2017, pp. 50–66.

[39] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space," in *IEEE Intelligent Vehicles Symposium*, 2019, pp. 2140–2147.

[40] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connected and Automated Vehicles Symposium*, 2020.

[41] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020.

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

[43] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[44] 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.

[45] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, pp. 1805–1824, 2000.

[46] 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 *21st International Conference on Intelligent Transportation Systems*. IEEE, 2018, pp. 2118–2125.

[47] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719 – 726.

[48] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[49] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *Proc. of the 19th International IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 485–491.

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

[51] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations*, 2015.

[52] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-Taliro: A tool for temporal logic falsification for hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2011, pp. 254–257.