Falsification-Based Robust Adversarial Reinforcement Learning

A PREPRINT

Xiao Wang, Saasha Nair, and Matthias Althoff*

July 20, 2020

ABSTRACT

Reinforcement learning (RL) has achieved tremendous progress in solving various sequential decision-making problems, e.g., control tasks in robotics. However, RL methods often fail to generalize to safety-critical scenarios since policies are overfitted to training environments. Previously, robust adversarial reinforcement learning (RARL) was proposed to train an adversarial network that applies disturbances to a system, which improves robustness in test scenarios. A drawback of neural-network-based adversaries is that integrating system requirements without handcrafting sophisticated reward signals is difficult. Safety falsification methods allow one to find a set of initial conditions as well as an input sequence, such that the system violates a given property formulated in temporal logic. In this paper, we propose falsification-based RARL (FRARL), the first generic framework for integrating temporal-logic falsification in adversarial learning to improve policy robustness. By using our falsification method, we do not need to construct an extra reward function for the adversary. We evaluate our approach on a braking assistance system and an adaptive cruise control system of autonomous vehicles. Experiments show that policies trained with a falsification-based adversary generalize better and show less violation of the safety specification in test scenarios than the ones trained without an adversary or with an adversarial network.

1 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 created a lot of interest in Reinforcement Learning (RL) [4]. Since then, RL has made great progress, e.g., in real-world applications such as robotics [5], natural language processing [6], and autonomous driving [7]. However, RL still suffers from shortcomings like bad generalization in real-world scenarios, risk-sensitive reward functions, and violation of safety constraints [8, 9]. This paper addresses the generalization problem caused by the huge amount of training required for RL.

Generalization is addressed by Pinto et al [10] by adding disturbances as adversarial examples [11], which was later extended by Pan et al. [12]. By training with adversarial examples, the authors reduce the simulation-to-reality gap caused by modeling errors, so that the trained models generalize better in real-world scenarios. Adversarial RL is formulated as a two-player zero-sum game in [10, 12], where 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 [13]. Otherwise, if we formulate the problem as a non-zero-sum game, in which the adversary optimizes a different reward function, a sophisticated reward function for the adversary would have to be handcrafted. One can argue that engineering the reward function could be helpful in improving the generalization ability to unseen scenarios of a RL agent, however, as pointed out in [9], designing the *perfect* reward function is itself a very challenging task. For instance, the traffic rule *a vehicle is not allowed to overtake another vehicle on its right side except in congested traffic* requires a sequence of events, which is difficult to integrate in a reward function, while the traffic rule can be easily expressed by temporal logic. Therefore, temporal

^{*}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

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 create adversarial samples in a single RL-agent setting, wherein the protagonist is represented as 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 [14].

The remainder of this paper is structured as follows: Section 2 provides an overview of current solutions for adversarial RL and system falsification for safety-critical systems. Section 3 introduces falsification-based RARL, which is evaluated in Section 4. We finish with conclusions and potential future research directions in Section 5.

2 RELATED WORK

2.1 Adversarial Reinforcement Learning

Despite the success of RL algorithms, they are susceptible to changes in environmental settings [9,15]. Hence, various forms of adversarial training [11] have been introduced to tackle this problem. One such approach involves adding adversarial perturbations to the observations of the agent, by attacking either only the image inputs [16–18], or addressing the entire state vector [19, 20]. Another approach involves the use of source-domain ensembles, which is adapted to the target domain using Bayesian model adaptation [21].

The approach most relevant to our work is the minimax approach extending Robust RL [15]. This approach, known as Robust Adversarial RL (RARL) [10], simultaneously trains two RL agents: one referred to as the protagonist and the other one referred to as the adversary. The adversary is tasked with applying destabilising forces to impede the protagonist, while the protagonist learns to be robust against the adversary. An extension of this work has been provided by Risk-Averse RARL (RARARL) [12], which focuses on safety-critical cyber-physical systems, by modeling the risk as the variance of an ensemble of value functions. In contrast to RARL and RARARL that model the setup as a two-agent reinforcement learning scenario, as mentioned in Section 1, our proposed solution reduces the number of reward functions to be defined and tuned, requires less parameters of the adversary to be optimized, as introduced in Section 3.1, and allows for better expressiveness for the adversary using temporal logic specifications.

2.2 Safety Falsification

Safety falsification methods aim to find an initial condition and input sequences, with which a system violates a given safety specification. Two categories of various approaches exist to tackle 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 found. Single-shooting is realized through Monte-Carlo techniques in [14,22,23], Ant-Colony Optimization in [24], Cross-Entropy method in [25], and Rapidly-exploring Random Tree search in [26–29]. A *multiple-shooting* approach is proposed in [30,31] to split system trajectories into small segments by simulating from multiple initial conditions in a state space decomposed into cells. Once one 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 Monte-Carlo Sampling are compared in [24] for two benchmarks and obtain similar results. The Cross-Entropy method outperfoms Monte-Carlo Sampling on five different benchmarks, as shown in [25]. Hence, we employ Cross-Entropy method in this work. Note that our framework can use other falsification approaches as well.

3 FALSIFICATION-BASED RARL

3.1 Safety Falsification

In order to formulate the safety falsification problem, we first introduce important definitions adopted from [14]. 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) [32]. 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 [33]. An atomic proposition p is a statement that can be either *true* or *false*. Atomic propositions and the logical connectives, e.g., Boolean operators *not* and *or* denoted by \neg and \lor , form propositional formulas. The temporal operator *until*, denoted by U, indicates that in a formula $\varphi_1 U \varphi_2$, 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*, denoted by \square , indicates that formula φ must hold for all times. MTL is an extension of temporal logic in which temporal

operators are replaced by time-constrained operators. So that U is replaced by U_I , where $I \subseteq (0, \infty)$, indicating that t is constrained by I. The syntax of an MTL formula φ is defined as follows [23]:

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

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 $a \implies b \equiv \neg a \lor b$. We use the *global* operator \square in this work as introduced 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 $x_0 \in \mathcal{X}_0$ as well as an input sequence $u: [0,T] \to \mathcal{U}$ such that the resulting trajectory y of system Σ violates the specification φ , which is denoted by

$$y(x_0, u) \not\models \varphi.$$
 (2)

Naïve falsification samples the set of initial conditions and input sequences uniformly. A more efficient approach is to guide the search with a metric measuring the distance between the trajectory and the set of states violating the specification. A *robustness metric* ε is proposed in [34] 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, 1 for successful falsification). The sign of ε reveals whether a trajectory y satisfies an MTL property φ . The robustness of y with respect to φ is denoted by

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

and defined as [23–25]:

$$[\![true]\!]_{d}(\boldsymbol{y},t) := +\infty$$

$$[\![p]\!]_{d}(\boldsymbol{y},t) := \mathbf{Dist}_{d}(\boldsymbol{y}(t),\mathcal{O}(p))$$

$$[\![\neg\varphi]\!]_{d}(\boldsymbol{y},t) := -[\![\varphi]\!]_{d}(\boldsymbol{y},t)$$

$$[\![\varphi_{1} \lor \varphi_{2}]\!]_{d}(\boldsymbol{y},t) := \max([\![\varphi_{1}]\!]_{d}(\boldsymbol{y},t), [\![\varphi_{2}]\!]_{d}(\boldsymbol{y},t))$$

$$[\![\varphi_{1}\boldsymbol{U}_{\boldsymbol{I}}\,\varphi_{2}]\!]_{d}(\boldsymbol{y},t) := \sup_{t' \in (t+\boldsymbol{I})} \min([\![\varphi_{2}]\!]_{d}(\boldsymbol{y},t'),$$

$$\inf_{t \leq t'' \leq t'} [\![\varphi_{1}]\!]_{d}(\boldsymbol{y},t'')),$$

$$(4)$$

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

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

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

$$d(y,x) = ||y - x||^2. (6)$$

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

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

The Cross-Entropy method combines piecewise-uniform distributions and Gaussian distributions to approximate the underlying distribution of the robustness value in (3) over the set $\mathcal{X}_0 \times \mathcal{U}$ [25]. The proposed distribution is denoted by p_θ with parameter θ , the unknown real distribution by q. The distance between two distributions is measured by the Kullback-Leibler Divergence [35]:

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

with $\xi \in \mathcal{X}_0 \times \mathcal{U}$. 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 . Those samples are sorted by their robustness values and the m least robust samples are taken, with $m \ll N_s$. Then, parameter θ is updated by minimizing the divergence $D(q, p_\theta)$ over m data samples. This procedure iterates until the divergence converges to a threshold. Subsequently, initial conditions and input sequences are sampled according to the converged distribution p_θ .

In this work, we consider an autonomous vehicle on a highway with two safety requirements: The agent is not allowed to collide with the leading vehicle and the agent is not allowed to drive backwards on the highway. We formulate these requirements as follows:

$$\Box(\neg\varphi_{\text{collision}} \land \neg\varphi_{\text{reverse}}). \tag{9}$$

Note that the proposed method can be directly applied to more complicated specifications containing temporal operators like *until* and *eventually*. In the future we will integrate more traffic rules in the system requirements as proposed by [36–38].

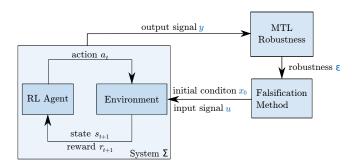


Figure 1: Falsification-based RARL framework. The RL agent and environment are regarded as the black box system Σ . Falsification model serves as the adversary, which provides input sets x_0 and u for the environment such that the system trajectory falsifies MTL specifications. The RL agent is trained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

3.2 Falsification-Based RARL

Figure 1 shows our framework of falsification-based RARL and Algorithm ?? 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 is the state space, A is the action space, P is the state transition probability, R is the expected reward signal, and $\gamma \in [0,1]$ is the discount factor. 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 [12] also observes. Therefore, we first train the agent for t_f time steps without an adversary, as shown in Algorithm ?? line 2-8, Fig. 3 and Fig. 4. Next, we regard our trained model and the environment as a black box system Σ . As shown in the right part of Fig. 1, we employ the previously-mentioned Cross-Entropy method to find 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 those initial conditions, change the behavior according to the new input sequences, and train the policy further in the new adversarial environment (line 13). This procedure repeats 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 \varphi; time step to start falsification t_f; number of falsification iterations n_f
   Initialize: Parameters of policy and value network \phi_0
   Result: Trained policy and value network \phi
   while t < T do
        for actor = 1, 2, ..., n_a do
 2
              Run policy \phi_{\text{old}} in E for t_a time steps;
 3
              Compute advantage estimates \hat{A}_1, ..., \hat{A}_{t_a} (10);
 4
 5
         Optimize surrogate L^{\text{CLIP+VF}} (12) wrt. \phi with batch size n_a t_a;
 6
         \phi_{\text{old}} \leftarrow \phi;
7
        t = t + n_a t_a ;
8
        if t > t_f then
 9
              Initialize falsifier parameter \theta_0;
10
              for iter = 1, 2, ..., n_f do
11
                   Sample input conditions x_0, u from p_{\theta_{\text{old}}}; Initialize new environment E_{\text{iter}} with x_0, change the behavior of E_{\text{iter}} with u;
12
13
                   Collect trajectories y in E_{\text{iter}} with agent \phi_{\text{old}} and evaluate robustness value according to (3);
14
                   Estimate (8) with y and minimize (8) wrt. \theta;
15
                   \theta_{\text{old}} \leftarrow \theta;
16
              end
17
              E \leftarrow E(\boldsymbol{x}_0, \boldsymbol{u});
18
19
         end
20
   end
```

We choose proximal policy optimization (PPO) [39] to optimize the policy network due to its superior performance in continuous control problems compared to other state-of-the-art approaches. We use an actor-critic architecture [40] to approximate both the policy and the value function with neural networks to reduce variance. We estimate the advantage function \hat{A}_t using a general advantage estimator (GAE) [41] as follows:

$$\hat{A}_t = \delta_t + (\gamma \lambda) \delta_{t+1} + \dots + \dots + (\gamma \lambda)^{T-t+1} \delta_{T-1},$$
with $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t),$ (11)

with
$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t),$$
 (11)

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

$$L^{\text{CLIP+VF}}(\phi) = \hat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\phi) - cL_t^{\text{VF}}(\phi) \right], \text{ with}$$
 (12a)

$$L^{\text{CLIP}}(\phi) = \hat{\mathbb{E}}_t \left[\min(r_t(\phi)\hat{A}_t, \text{clip}(r_t(\phi), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right]$$
(12b)

$$L_t^{\text{VF}}(\phi) = (V_\phi(s_t) - V_t^{\text{targ}})^2, \tag{12c}$$

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

EXPERIMENTS

We evaluate our approach on two systems. The first one is a braking assistance (BA) system of an autonomous vehicle to avoid rear-end collisions and avoid driving reversely on a highway. The second one is an adaptive cruise control (ACC) system that keeps a safe distance to the leading vehicle and follows a desired velocity. The two systems are implemented in the same traffic simulator with different reward functions described in Section 4.2. To fairly evaluate the performance of our approach, we train each system in three different environments: a baseline environment without an adversarial model, an adversarial environment with an RL agent as adversary, and an adversarial environment with an adversary using our falsification method. The adversarial RL agent in the second environment is trained using RARL [10]. We choose RARL over the more recent RARARL [12] to train the adversarial RL agent because RARARL was proposed to tackle discrete action space. RARARL can not be directly applied to problems with a continuous action space. We call the policy that controls the ego vehicle the protagonist as [10] does.

4.1 Dataset

In the adversarial environments, the behavior of the leading vehicle is altered by the falsification method or an adversarial RL agent. The baseline environment could be realized by either rule-based driver models, e.g., the intelligent driver model (IDM) [42], or real traffic data. A limitation of rule-based driver models is their homogeneity. The policy could easily overfit to reacting 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 HighD dataset of naturalistic vehicle trajectories on German highways [43].

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

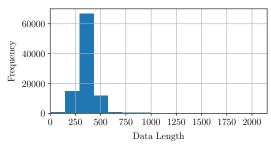


Figure 2: Frequency histogram of the length of lane-following trajectories in HighD dataset [43].

4.2 Environment

We set up a driving simulator based on the CommonRoad benchmark suite [44] and OpenAI Gym [45]. Since the goal of the agent is to learn the longitudinal driving behavior, our simulator contains only a straight lane of $600\,\mathrm{m}$ length. 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 according to a point-mass model, whose input is acceleration, which is sampled from the policy network. In order to assure 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 both vehicles have the same maximum deceleration $a_{\mathrm{max}} = 10\,\mathrm{m/s^2}$. The safe distance is then computed according to [46] as

$$s_{\text{safe}} = \frac{1}{2a_{\text{max}}} (v_f^2 - v_l^2) + v_f \delta, \tag{13}$$

where v_f and v_l are the velocities of the following and leading vehicles respectively, and δ is the reaction delay of the following vehicle. We use $\delta = 0.3 \, \text{s}$, as assumed by [46] for autonomous vehicles.

Without loss of generality, the initial position of the ego vehicle is fixed at $s_{\rm ego}=10\,\rm m$, whereas the initial position of the leading vehicle is either randomly sampled within the range $[s_{\rm ego}+s_{\rm safe},s_{\rm ego}+s_{\rm safe}+40]$ or calculated by the falsification tool. Acceleration and initial velocity of the leading vehicle are either extracted from the selected HighD trajectories, or computed by the adversaries, i.e., in (7) x_0 corresponds to the initial position and velocity of the leading vehicle and u corresponds to the acceleration of the leading vehicle.

Since maintaining a safe distance to the leading vehicle is crucial for collision avoidance, the feature vector of the policy networks needs to provide all necessary information to calculate the safe distance in (13). Thus, we choose the feature vector for both systems as listed in Tab. 1.

The reward function of the protagonist of the braking assistance (BA) system is defined as:

$$r_{\rm BA} = \begin{cases} -1, & \text{if agent drives reversely or collision happens} \\ 0, & \text{otherwise.} \end{cases}$$
 (14)

The reward function of the protagonist of the adaptive cruise control (ACC) system is defined as:

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

$$(15)$$

Table 1: Features used by policy network

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

The first and last items are the same as in $r_{\rm BA}$. Two additional terms are added: the second term penalizes a violation of the safe distance with a nonlinear function which 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 (14) and (15) are selected by a grid search. The nonlinear functions in the second and third terms in (15) increase the performance of the agent significantly.

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

4.3 Baseline Model

As mentioned in Section 3.2, we train our policies and value functions with PPO [39] and an actor-critic algorithm [40]. In particular, we use a shared network design to share features between the policy and the value function. We build our models on top of the OpenAI Baselines implementation [47]. Since our goal is to compare all methods with the same hyper-parameters, we did not perform an hyper-parameter optimization for each method. Instead, we use the default hyper-parameters that the OpenAI Baselines implementation [47] provides. The shared policy and value network has two hidden layers with 64 neurons each and *tanh* as activation function. The model is optimized using the Adam optimizer [48] with a learning rate of 0.0003 and a batch size of 128. The discount factor of the advantage estimator in (10) is $\lambda = 0.95$.

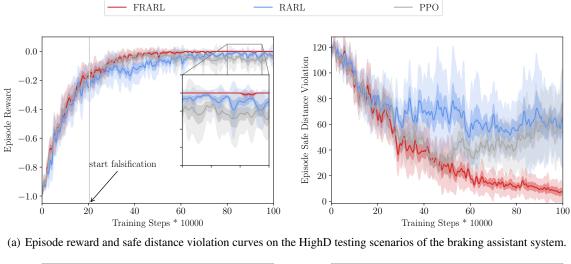
In all experiments, we first train the protagonist policy without the adversary for 200,000 training steps to let it learn basic skills as suggested in [12]. In the RL adversarial environment, we update the parameters of the protagonist θ_{μ} to maximize $r_{\rm BA}$ or $r_{\rm ACC}$ for $N_{\mu}=10$ iterations while the parameters of the adversary θ_{ν} are kept constant. Then we hold θ_{μ} constant and update θ_{ν} for $N_{\nu}=1$ iteration to maximize $r_{\rm adv}$. N_{μ} and N_{ν} are chosen empirically. This process iterates until both policies converge. In the falsification adversarial environment, we apply S-Taliro [49] to falsify the protagonist during training, which is a MATLAB toolbox for MTL falsification for hybrid systems. S-Taliro is called every 10 iterations to compute 10 acceleration traces and initial positions and velocities for the leading vehicle, with which the protagonist falsifies the given specification (9). The computed traces are randomly picked by the simulator to train the policy further. In the following part, we call the baseline model PPO, the policy model trained with a RL adversarial agent RARL, and the policy model trained with our method FRARL.

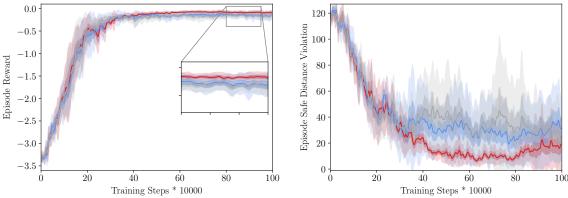
4.4 Evaluation

During the training phase, policies are set to be stochastic to encourage exploration, whereas during the evaluation as well as the falsification phase, deterministic policies are used. In order 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 test scenarios and random test scenarios, where the acceleration of the leading vehicle is randomly sampled within a given range. Note that we use 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 random test scenarios. In order to analyze the safety of the behavior of the agent, we count the number of time steps when the agent violates the safe distance to the leading vehicle. Figure 3 and Figure 4 show the episode reward and number of safe distance violations in the random and highD test scenarios of the braking assistance and adaptive cruise control system trained with PPO, RARL, and FRARL. For both systems, FRARL shows slightly higher episode reward and much less safe distance violations. In addition, for the braking assistance system, FRARL converges to zero reward already at half of the training steps. Furthermore, FRARL has lower variance for the episode reward and safe distance violations. It also shows a better advanrage rate in random test scenarios, which indicates that training in a falsified environment improves the ability of a RL agent to generalize to unknown scenarios.

To further address the robustness of the trained models, we evaluate all models on 28 037 HighD test scenarios as well as random scenarios and show the average rate of reverse driving and collisions in Tab. 2 and Tab. 3. For both systems on both test scenarios, FRARL achieves the lowest rate of collisions and reverse driving. The reason that FRARL outperforms RARL is that an adversarial RL agent seeks scenarios where 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. Instead, our falsification method optimizes the scenarios until safety-critical scenarios are found. Therefore, the policies trained in safety-critical scenarios behave much safer than the classically-trained policies.





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

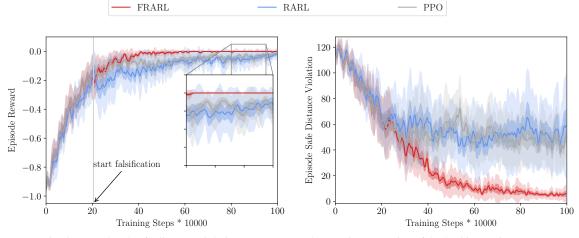
Figure 3: The learning curves of episode reward and number of safe distance violations on the **HighD** test scenarios of the braking assistance and adaptive cruise control system trained with PPO, RARL, and FRARL. For both systems, FRARL shows slightly higher episode reward and much less safe distance violations. In addition, for the braking assistance system, FRARL converges to zero reward already at half of the training steps. Furthermore, FRARL has lower variance for the episode reward and safe distance violations.

T 11 0 4	D CIT C	D 1 '	00.00FIT 1.D.	
Table 2: Average	Rate of Unsafe	Behaviors over	28 037 HighD to	est scenarios

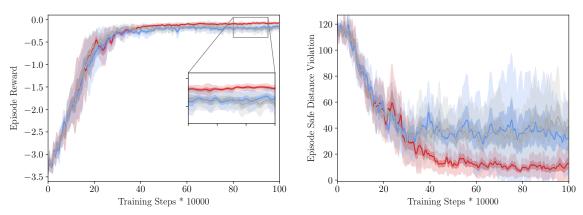
	BA		ACC	
Violation	Reverse	Collision	Reverse	Collision
PPO	0.34%	4.59% 2.70%	0.005%	0.24%
RARL	0.009%	2.70%	0	0.17%
FRARL	0	0.015%	0	0

5 CONCLUSIONS

We present a framework for combining reinforcement learning (RL) with safety falsification methods, which serves as adversarial RL, in order to improve the robustness of trained policies. By formulating safety requirements in metric temporal logics (MTL), we spare ourselves the trouble of handcrafting a reward function for the adversary. We demonstrate for a braking assistance system and an adaptive cruise control system that the policies trained with our approach satisfy the safety specification much better in test scenarios, and thus are more robust. In the future, we will



(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.

Figure 4: The learning curves of episode reward and number of safe distance violations on the **random** test scenarios of the braking assistance and adaptive cruise control system trained with PPO, RARL, and FRARL FRARL show more advantage on random test scenarios than on the HighD scenarios, which indicates that training in a falsified environment improves the ability of a RL agent to generalize to unknown scenarios.

Table 3: 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

extend our experiments to more complex driving scenarios, e.g., urban scenarios. Moreover, we will integrate traffic rules in our MTL specifications.

ACKNOWLEDGMENT

The authors gratefully acknowledge the partial financial support of this work by the German Research Foundation Grant AL 1185/3-2 and the Ford Motor Company.

References

- [1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [3] Tanguy Chouard. The Go Files: AI computer wraps up 4-1 victory against human champion. *Nature News*, 2016.
- [4] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.
- [5] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):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*, 31(7):2469–2489, 2020.
- [7] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.
- [8] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.
- [9] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [10] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proc. of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826, 2017.
- [11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR*, 2014.
- [12] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, pages 8522–8528. IEEE, 2019.
- [13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in neural information processing systems*, pages 2234–2242, 2016.
- [14] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems* (*TECS*), 12(2s):1–30, 2013.
- [15] Jun Morimoto and Kenji Doya. Robust reinforcement learning. Neural computation, 17(2):335–359, 2005.
- [16] Sandy Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In 5th International Conference on Learning Representations, ICLR, 2017.
- [17] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *Proc. of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 3756–3762.
- [18] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. In 5th International Conference on Learning Representations, ICLR, 2017.
- [19] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3932–3939. IEEE, 2017.
- [20] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2040–2042, 2018.
- [21] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. In 5th International Conference on Learning Representations, ICLR, 2017.

- [22] Houssam Abbas and Georgios Fainekos. Linear hybrid system falsification through local search. In *International Symposium on Automated Technology for Verification and Analysis*, pages 503–510, 2011.
- [23] Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivancić, Aarti Gupta, and George 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*, pages 211–220. ACM, 2010.
- [24] Yashwanth Singh Rahul Annapureddy and Georgios E Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, pages 91–96. IEEE, 2010.
- [25] Sriram Sankaranarayanan and Georgios 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*, pages 125–134. ACM, 2012.
- [26] Amit Bhatia and Emilio Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 142–156, 2004.
- [27] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin, and Jyotirmoy V Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods Symposium*, pages 127–142, 2015.
- [28] Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. Compositional falsification of cyber-physical systems with machine learning components. In *NASA Formal Methods Symposium*, pages 357–372, 2017.
- [29] Markus Koschi, Christian Pek, Sebastian Maierhofer, and Matthias Althoff. Computationally efficient safety falsification of adaptive cruise control systems. In *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2019.
- [30] Aditya Zutshi, Sriram Sankaranarayanan, Jyotirmoy V Deshmukh, and James Kapinski. A trajectory splicing approach to concretizing counterexamples for hybrid systems. In *52nd Conference on Decision and Control*, pages 3918–3925. IEEE, 2013.
- [31] Aditya Zutshi, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and James Kapinski. Multiple shooting, CEGAR-based falsification for hybrid systems. In *Proc. of the 14th International Conference on Embedded Software*, pages 1–10. ACM, 2014.
- [32] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.
- [33] E Allen Emerson. Temporal and modal logic. In Formal Models and Semantics, pages 995–1072. 1990.
- [34] Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.
- [35] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [36] Albert Rizaldi and Matthias Althoff. Formalising traffic rules for accountability of autonomous vehicles. In 2015 *IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1658–1665, 2015.
- [37] Albert Rizaldi, Jonas Keinholz, Monika Huber, Jochen Feldle, Fabian Immler, Matthias Althoff, Eric Hilgendorf, and Tobias Nipkow. Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. In *International Conference on Integrated Formal Methods*, pages 50–66, 2017.
- [38] Klemens Esterle, Vincent Aravantinos, and Alois Knoll. From specifications to behavior: Maneuver verification in a semantic state space. In *IEEE Intelligent Vehicles Symposium*, pages 2140–2147, 2019.
- [39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [40] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.
- [41] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In 4th International Conference on Learning Representations, ICLR, 2016.
- [42] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805–1824, 2000.

- [43] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz 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, pages 2118–2125. IEEE, 2018.
- [44] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 719 726, 2017.
- [45] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [46] 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*, pages 485–491, 2016.
- [47] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI Baselines. https://github.com/openai/baselines, 2017.
- [48] D Kingma and J Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.
- [49] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram 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*, pages 254–257, 2011.