Targeted Adversarial Attacks on Deep Reinforcement Learning Policies via Model Checking

Dennis Gross[1], Thiago D. Simão[1], Nils Jansen[1] and Guillermo A. Pérez[2]
[1]*Institute for Computing and Information Sciences, Radboud University, Toernooiveld 212, 6525 EC Nijmegen, The Netherlands*
[2]*Department of Computer Science, University of Antwerp – Flanders Make, Middelheimlaan 1, 2020 Antwerpen, Belgium*

Keywords: Adversarial Reinforcement Learning, Model Checking.

Abstract: Deep Reinforcement Learning (DRL) agents are susceptible to adversarial noise in their observations that can mislead their policies and decrease their performance. However, an adversary may be interested not only in decreasing the reward, but also in modifying specific temporal logic properties of the policy. This paper presents a metric that measures the exact impact of adversarial attacks against such properties. We use this metric to craft optimal adversarial attacks. Furthermore, we introduce a model checking method that allows us to verify the robustness of RL policies against adversarial attacks. Our empirical analysis confirms (1) the quality of our metric to craft adversarial attacks against temporal logic properties, and (2) that we are able to concisely assess a system's robustness against attacks.

1 INTRODUCTION

Deep reinforcement learning (DRL) has changed how we build agents for sequential decision-making problems (Mnih et al., 2015). It has triggered applications in critical domains like energy and transportation (Farazi et al., 2021; Nakabi and Toivanen, 2021). An RL agent learns a near-optimal policy (based on a given objective) by making observations and gaining rewards through interacting with the environment (Sutton and Barto, 2018). Despite the success of RL, potential security risks limit its usage in real-life applications. The so-called adversarial attacks introduce noise into the observations and mislead the RL decision-making to drop the cumulative reward, which may lead to unsafe behaviour (Huang et al., 2017; Amodei et al., 2016).

Generally, rewards lack the expressiveness to encode complex safety requirements (Vamplew et al., 2022; Hasanbeig et al., 2020). Therefore, for an adversary, capturing how much the cumulative reward is reduced may be too generic for attacks targeting specific safety requirements. For instance, an RL taxi agent may be optimized to transport passengers to their destinations. With the already existing adversarial attacks, the attacker can prevent the agent from transporting the passenger. However, the attacker cannot create controlled adversarial attacks that may increase the probability that the passenger never gets picked up or that the passenger gets picked up but never arrives at its destination. More generally, current adversary attacks are not able to control temporal logic properties.

This paper aims to combine adversarial RL with rigorous model checking (Baier and Katoen, 2008), which allows the adversary to create so-called *property impact attacks (PIAs)* that can influence specific RL policy properties. These PIAs are not limited by properties that can be expressed by rewards (Hahn et al., 2019; Hasanbeig et al., 2020; Vamplew et al., 2022), but support a broader range of properties that can be expressed by *probabilistic computation tree logic* (PCTL; Hansson and Jonsson, 1994). Our experiments show that for PCTL properties, it is possible to create targeted adversarial attacks that influence them specifically. Furthermore, the combination of model checking and adversarial RL allows us to verify via *permissive policies* (Dräger et al., 2015) how vulnerable trained policies are against PIAs. Our *main contributions are:* a metric to measure the impact of adversarial attacks on a broad range of RL policy properties, a property impact attack (PIA) to target specific properties of a trained RL policy, and a method that checks the robustness of RL policies against adversarial attacks.
The empirical analysis shows that the method to at-

Gross, D., Simão, T., Jansen, N. and Pérez, G.
Targeted Adversarial Attacks on Deep Reinforcement Learning Policies via Model Checking.
DOI: 10.5220/0011693200003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 501-508
ISBN: 978-989-758-623-1; ISSN: 2184-433X

tack RL policies can effectively modify PCTL properties. Furthermore, the results support the theoretical claim that it is possible to model check the robustness of RL policies against property impact attacks.

The paper is structured in the following way. First, we summarize the related work and position our paper in it. Second, we explain the fundamentals of our technique. Then, we present the adversarial attack setting, define our property impact attack, and show a way to model check policy robustness against such adversarial attacks. After that, we evaluate our methods in multiple environments.

2 RELATED WORK

We now summarize the related work and position our paper in between adversarial RL and model checking.

There exist a variety of adversarial attack methods to attack RL policies with the goal of dropping their total expected reward (Chan et al., 2020; Lin et al., 2017b; Ilahi et al., 2022; Lin et al., 2017a; Clark et al., 2018; Yu and Sun, 2022). The first proposed adversarial attack on DRL policies (Huang et al., 2017) uses a modified version of the *fast gradient sign method (FGSM)*, developed by Goodfellow et al. (2015), to force the RL policy to make malicious decisions (for more details, see Section 3.2). However, none of the previous work let the attacker target temporal logic properties of RL policies. Chan et al. (2020) create more effective attacks that modify only one feature (if the smallest sliding window is used) of the agent's observation by empirically measuring the impact of each feature on the reward. We build upon this idea to measure the feature impact on temporal logic properties.

3 BACKGROUND

In this section, we introduce the necessary foundations.

3.1 Probabilistic Systems

A *probability distribution* over a set X is a function $\mu : X \to [0,1]$ with $\sum_{x \in X} \mu(x) = 1$. The set of all distributions over X is denoted by $Distr(X)$.

Definition 3.1 (Markov Decision Process)**.** A Markov decision process (MDP) is a tuple $M = (S, s_0, Act, T, rew)$ where S is a finite, nonempty set of states, $s_0 \in S$ is an initial state, Act is a finite set of actions, $T\colon S \times Act \to Distr(S)$ is a probability transition function. We employ a factored state representation $S \subseteq \mathbb{Z}^n$, where each state $s \in \mathbb{Z}^n$ is an n-dimensional vector of features $(f_1, f_2, ..., f_n)$ such that $f_i \in \mathbb{Z}$ for $1 \leq i \leq n$. We define $rew\colon S \times Act \to \mathbb{R}$ as a reward function.

The available actions in $s \in S$ are $Act(s) = \{a \in Act \mid T(s,a) \neq \bot\}$. An MDP with only one action per state ($\forall s \in S\colon |Act(s)| = 1$) is a discrete-time Markov chain (DTMC). Note that features do not necessarily have to have the same domain size. We define $\mathcal{F}$ as the set of all features f_i in state $s \in S$.

A path of an MDP M is an (in)finite sequence $\tau = s_0 \xrightarrow{a_0, r_0} s_1 \xrightarrow{a_1, r_1} ...$, where $s_i \in S$, $a_i \in Act(s_i)$, $r_i := rew(s_i, a_i)$, and $T(s_i, a_i)(s_{i+1}) \neq 0$. A state s' is reachable from state s if there exists a path τ from state s to state s'. We say a state s is reachable if s is reachable from s_0.

Definition 3.2 (Policy)**.** A memoryless deterministic policy for an MDP $M{=}(S, s_0, Act, T, rew)$ is a function $\pi\colon S \to Act$ that maps a state $s \in S$ to an action $a \in Act(s)$.

Applying a policy π to an MDP M yields an *induced DTMC*, denoted as D, where all nondeterminism is resolved. We say a state s is reachable by a policy π if s is reachable in the DTMC induced by π. Λ is the set of all possible memoryless policies.

To analyze the properties of an induced DTMC, it is necessary to specify the properties via a specification language like probabilistic computation tree logic PCTL (Hansson and Jonsson, 1994).

Definition 3.3 (PCTL Syntax)**.** Let AP be a set of atomic propositions. The following grammar defines a state formula: $\Phi := \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid P_{\bowtie p}[\phi] \mid P^{max}_{\bowtie p}(\phi) \mid P^{min}_{\bowtie p}(\phi)$ where $a \in AP, \bowtie \in \{<, >, \leq, \geq\}$, $p \in [0,1]$ is a threshold, and ϕ is a path formula which is formed according to the following grammar $\phi := X\Phi \mid \phi_1\ U\ \phi_2 \mid \phi_1\ F_{\theta t}\ \phi_2 \mid G\ \Phi$ with $\theta = \{<, \leq\}$.

PCTL formulae are interpreted over the states of an induced DTMC. In a slight abuse of notation, we use PCTL state formulas to denote probability values. That is, we sometimes write $P_{\bowtie p}(\phi)$ where we omit the threshold p. For instance, $P(F_{\leq 100} collision)$ denotes the reachability probability of eventually running into a collision within the first 100 time steps.

There is a variety of model checking algorithms for verifying PCTL properties (Courcoubetis and Yannakakis, 1988, 1995), and PRISM and Storm offer efficient and mature tool support (Kwiatkowska et al., 2011; Hensel et al., 2022). COOL-MC (Gross et al., 2022a) allows model checking of a trained RL policy against a PCTL property and MDP. The tool builds the

induced DTMC on the fly via an *incremental building process* (Cassez et al., 2005; David et al., 2015).

3.2 Adversarial Attacks on DRL Policies

The standard learning goal for RL is to find a policy π in a MDP such that π maximizes the expected accumulated discounted rewards, that is, $\mathbb{E}[\sum_{t=0}^{L} \gamma^t R_t]$, where γ with $0 \leq \gamma \leq 1$ is the discount factor, R_t is the reward at time t, and L is the total number of steps. DRL uses neural networks to train policies. A neural network is a function parameterized by weights θ. In DRL, the policy π is encoded using a neural network which can be trained by minimizing a sequence of loss functions $J(\theta, s, a)$ (Mnih et al., 2013).

An *adversary* is a malicious actor that seeks to harm or undermine the performance of an RL system. For instance, an adversary may try to decrease the expected discounted reward by attacking the RL policy via adversarial attacks.

Definition 3.4 (Adversarial Attack)**.** An *adversarial attack* $\delta\colon S \to S$ maps a state s to an *adversarial state* s_{adv} (see Figure 1). A successful adversarial attack at a given state s leads to a *misjudgment* of the RL policy ($\pi(s) \neq \pi(\delta(s))$) and an attack is ε-bounded if $\|\delta(s) - s\|_\infty \leq \varepsilon$ with l_∞-norm defined as $\|\delta(s) - s\|_\infty = max_{\delta_i \in \delta} |\delta_i - s_i|$.

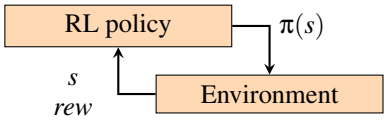
Recall that states are n-dimensional vectors of features from $\mathbb{Z}^n$. Executing a policy π on an MDP M and attacking the policy π at each reachable state s by δ yields an adversarial-induced DTMC D_{adv}. There exist a variety of adversarial attack methods to create adversarial attacks δ (Ilahi et al., 2022; Gleave et al., 2020; Lee et al., 2020, 2021; Rakhsha et al., 2020; Carlini and Wagner, 2017).

Our work builds upon the FGSM attack and the work of Chan et al. (2020). Given the weights θ of the neural network policy π and a loss $J(\theta, s, a)$ with state s and $a \coloneqq \pi(s)$, the FGSM, denoted as $\delta_{\text{FGSM}}\colon S \to S$, adds noise whose direction is the same as the gradient of the loss $J(\theta, s, a)$ w.r.t the state s to the state s (Huang et al., 2017) and the noise is scaled by $\varepsilon \in \mathbb{Z}$ (see Equation (1)). Note that we are dealing with integer ε-values because our states are comprised of integer features. We specify the $\bigtriangledown$-operator as a vector differential operator. Depending on the gradient, we either add or subtract ε.

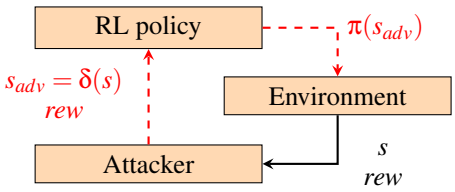
$$\delta_{\text{FGSM}}(s) = s + \varepsilon \cdot sign(\bigtriangledown_s J(\theta, s, a)) \quad (1)$$

A FGSM for feature f_i, denoted as $\delta^{(f_i)}_{\text{FGSM}}(s)$, modifies only the feature f_i in state s.

$$\delta^{(f_i)}_{\text{FGSM}}(s) = s + \varepsilon \cdot sign(\bigtriangledown_{s_{f_i}} J(\theta, s, a)) \quad (2)$$



(a) RL policy interaction with the environment.



(b) An adversary manipulates with δ the observations of the RL policy π and its interaction with the environment.

Figure 1: RL (a) vs. adversarial RL (b).

We denote the set of all possible ε-bounded attacks at state s via feature f_i, including $\delta^{(f_i)}(s) = s$ for no attack, as $\Delta^{(f_i)}_{\varepsilon}(s)$.

Chan et al. (2020) first generate for all features a static reward impact (SRI) map by attacking each feature (in the case of the smallest sliding window) with the FGSM attack to measure its impact (the drop of the expected reward) offline. A feature f_i with a more significant impact indicates that changing this feature f_i via $\delta^{(f_i)}_{\text{FGSM}}$ will influence the expected discounted reward more than via another feature f_k with a less significant impact. For each feature f_i, this is done multiple times N, where each iteration executes the RL policy on the environment and attacks at every state the feature f_i via the FGSM attack $\delta^{(f_i)}_{\text{FGSM}}$. After calculating the SRI, they use all the SRI values of the features f_i to select the most vulnerable feature to attack the deployed RL policy.

Adversarial training retrains the already trained RL policy by using adversarial attacks during training to increase the RL policy robustness (Pinto et al., 2017; Liu et al., 2022; Korkmaz, 2021b).

4 METHODOLOGY

We introduce the general adversarial setting, the property impact (PI), the property impact attack (PIA), and bounded robustness.

4.1 Attack Setting

We first describe our method's adversarial attack setting (adversary's goals, knowledge, and capabilities).

Goal. The adversary aims to modify the prop-

erty value of the target RL policy π in its environment (modeled as an MDP). For instance, the adversary may try to increase the probability that the agent collides with another object (i.e. $\max_{\delta} P(F\ collision)$ in the adversarial-induced DTMC).

Knowledge. The adversary that knows the weights θ of the trained policy (for the FGSM attack) and knows the MDP of the environment. Note that we can replace the FGSM attack with any other attack. Therefore, knowing the weights of the trained policy should not be a strict constraint.

Capabilities. The adversary can attack the trained policy π at every visited state s during the incremental building process for the model checking of the adversarial-induced DTMC and after the RL policy is deployed.

4.2 Property Impact Attack (PIA)

Combining adversarial RL with model checking allows us to craft adversarial property impact attacks (PIAs) that target temporal logic properties. Our work builds upon the research of Chan et al. (2020). Instead of calculating SRIs (see Section 3.2), we calculate property impacts (PIs). The PI values are used to select the feature f_i with the most significant PI-value to attack the deployed RL policy in its environment ($f_i = \text{argmax}_{f_i \in \mathcal{F}} PI(\pi, P(\phi), f_i, \varepsilon)$).

Definition 4.1 (Property Impact)**.** The *property impact* $PI\colon \Lambda \times \Theta \times \mathcal{F} \times \mathbb{Q} \to \mathbb{Q}$ quantifies the impact of an adversarial attack $\delta^{(f_i)}_{\text{FGSM}} \in \Delta^{(f_i)}_{\varepsilon}(s)$ via a feature $f_i \in \mathcal{F}$ on a given RL policy property $P(\phi) \in \Theta$ with Θ as the set of all possible PCTL properties for the MDP M.

A feature f_i with a more significant PI-value indicates that changing this feature f_i via $\delta^{(f_i)}_{\text{FGSM}}$ will influence the property (expressed by the property query $P(\phi)$) more than via another feature f_k with a less significant PI-value.

We now explain how to calculate the PI-value for a given MDP M, policy π, PCTL property query $P(\phi)$, feature f_i, and FGSM attack $\delta^{(f_i)}_{\text{FGSM}}$. First, we incrementally build the induced DTMC of the policy π and the MDP M to check the property value r of the policy π. We do this by using COOL-MC and inputting the MDP M, policy π, and PCTL property query $P(\phi)$ into it to calculate the probability r. Second, we incrementally build the adversarial-induced DTMC D_{adv} of the policy π and the MDP M with the ε-bounded FGSM attack $\delta^{(f_i)}_{\text{FGSM}}$ to check its probability r_{adv}. To support the building and model checking of adversarial-induced DTMCs via *adv_property_result*, we extend the incremental building process of COOL-MC in the following way. For every reachable state s by the policy π, the policy π is queried for an action $a = \pi(s)$. In the underlying MDP, only states s that may be reached via that action a are expanded. The resulting model is fully probabilistic, as no action choices are left open. It is, in fact, the Markov chain induced by the original MDP M and the policy π. An adversary can now inject adversarial attacks $\delta(s)$ at every state s that gets passed to the policy π during the incrementally building process (Zhang et al., 2020). This may lead to the effect that the policy π makes a misjudgment ($\pi(s) \neq \pi(\delta(s))$ and results into an adversarial-induced DTMC D_{adv}. This allows us to model check the adversarial-induced DTMCs D_{adv} to gain the adversarial probability r_{adv}. Finally, we measure the property impact value by measuring the absolute difference between r and r_{adv}.

4.3 RL Policy Robustness

A trained RL policy π can be robust against an ε-bounded PIA that attacks a temporal logic property $P(\phi)$ via feature f_i ($PI(\pi, P(\phi), f_i, \varepsilon) = 0$). However, this is a weak statement about robustness since there still exist multiple adversarial attacks $\delta^{(f_i)}(s)$ with $\|\delta^{(f_i)}(s) - s\|_\infty \leq \varepsilon$ generated by other attacks, such as the method from Carlini and Wagner (2017).

Given a fixed policy π and a set of attacks $\Delta^{(f_i)}_{\varepsilon}(s)$, we generate a *permissive policy* Ω. Applying this policy π in the original MDP M generates a new MDP M' that describes all potential behavior of the agent under the attack.

Definition 4.2 (Behavior under attack)**.** A permissive policy $\Omega\colon S \to 2^{\mathsf{Act}}$ selects, at every state s, all actions that can be queried via $\Delta^{(f_i)}_{s}(s)$. We consider $\Omega(s) = \bigcup_{\delta^{(f_i)}_i \in \Delta^{(f_i)}_s(s)} \pi(\delta^{(f_i)}_i(s))$ with $\pi(\delta^{(f_i)}_i(s)) \in Act(s)$.

Applying a permissive policy to an MDP does not necessarily resolve all nondeterminism, since more than one action may be selected in some state(s). The induced model is then (again) an MDP. We are able to apply model checking, which typically results in best- and worst-case probability bounds $P^{max}(\phi)$ and $P^{min}(\phi)$ for a given property query $P(\phi)$.

We use the induced MDP to model check the *robustness* (see Definition 4.3) against every possible ε-bounded attack $\delta^{(f_i)}(s)$ for a trained RL policy π in its environment and bound the robustness to an α-threshold (property impacts below a given threshold α may be acceptable).

Definition 4.3 (Bounded robustness)**.** A policy π is called *robustly bounded* by ε and α (ε, α-robust) for

property query ϕ if it holds that

$$|P^*(\phi) - P(\phi)| \leq \alpha \tag{3}$$

for all possible ε-bounded adversarial attacks $\Delta_\varepsilon^{(f_i)}(s)$ at every reachable state s by the permissive policy Ω. We define $\alpha \in \mathbb{Q}$ as a threshold (in this paper, we focus on probabilities and therefore $\alpha \in [0,1]$). $|P^*(\phi) - P(\phi)|$ stands for the largest impact of a possible attack. We denote P^* as P^{max} or P^{min} depending if the attack should increase (P^{max}) or decrease (P^{min}) the probability.

By model checking the robustness of the trained RL policies (as described in Section 4.3), it is possible to extract for each state s the adversarial attack $\delta^{(f_i)}$ that is part of the most impactful attack and use the corresponding attack as soon as the state gets observed by the adversary. This is possible because the underlying model of the induced MDP allows the extraction of the state and action pairs (s, a_{adv}) that lead to the wanted property value modification ($a_{adv} := \pi(\delta^{(f_i)}(s))$).

5 EXPERIMENTS

We now evaluate our PI method, property impact attack (PIA), and robustness checker method in multiple environments. The experiments are performed by initially training the RL policies using the deep Q-learning algorithm (Mnih et al., 2013), then using the trained policies to answer our research questions.

5.1 Setup

We now explain the setup of our experiments.

Environments. We used our proposed methods in a variety of environments (see Figure 2, Figure 4, and Table 2). We use the *Freeway* (for a fair comparison between the SRI and PI method) and the *Taxi* environment. Additionally, we use the environments *Collision Avoidance*, *Stock Market*, and *Smart Grid* (see Gross et al. (2022b) for more details).

Freeway is an action video game for the Atari 2600. A player controls a chicken (up, down, no operation) who must run across a highway filled with traffic to get to the other side. Every time the chicken gets across the highway, it earns a reward of one. An episode ends if the chicken gets hit by a car or reaches the other side. Each state is an image of the game's state. Note that we use an abstraction of the original game (see Figure 2).

In the *Taxi* environment, the agent must pick up passengers and transport them to their destination

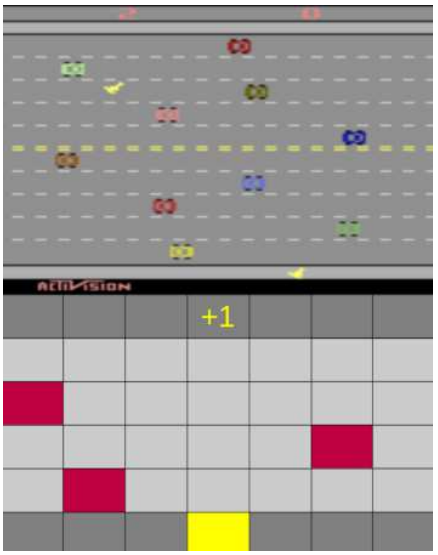


Figure 2: A comparison between the Atari 2600 Freeway game (top) and our abstracted version (bottom).

without running out of fuel. The environment ends when the agent completes a predefined number of jobs or runs out of fuel. The maximal fuel level for the taxi is ten and the maximal number of jobs is two. The agent can refuel at the gas station cell ($x = 1, y = 2$). The problem is formalized as follows:

$$S = \{(x, y, Xloc, Yloc, Xdest, Ydest, fuel, done, pass, jobs, done), ...\}$$

$$Act = \{north, east, south, west, pick\_up, drop\}$$

$$penalty = \begin{cases} 0, \text{ if passenger successfully dropped.} \\ 21, \text{ if passenger got picked up.} \\ 21 + |x - Xdest| + \\ \quad |y - Ydest|, \text{ if passenger on board.} \\ 21 + |x - Xloc| + \\ \quad |y - Yloc|, \text{ if passenger not on board.} \\ 1500, \text{ if not at gas station and out of fuel.} \end{cases}$$

Properties. Table 1 presents the property queries of the policy trained by an RL agent achieves in these properties without the attack (=).

Trained RL Policies. We trained in a standard way using COOL-MC (Gross et al., 2022a).

Technical Setup. All experiments were executed on an NVIDIA GeForce GTX 1060 Mobile GPU, 16 GB RAM, and an Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz x 12. For model checking, we use Storm 1.7.1 (dev).

5.2 Analysis

We now answer our research questions.

Table 1: PCTL property queries, with their labels and the original result of the property query without an attack (=). *Fr* stands for *Freeway*, *Coll.* stands for *Collision Avoidance*, *SG* for *Smart Grid*, and *SM* for *Stock Market*.

Env.	**Label**	**PCTL Property Query** ($P(\phi)$)	=
Fr	crossed	$P(F\ crossed)$	1.0
Taxi	deadlock1	$P(fuel \geq 4\ U\ (G(jobs = 1 \wedge \neg empty\ \wedge pass)))$	0.0
	deadlock2	$P(fuel \geq 4\ U\ (G(jobs = 1 \wedge \neg empty \wedge \neg pass)))$	0.0
	station_empty	$P((((jobs{=}0\ U\ x{=}1 \wedge y{=}2)\ U\ (jobs{=}0 \wedge \neg(x{=}1 \wedge y{=}2)))\ U\ empty \wedge jobs{=}0))$	0.0
	$\overline{\text{station}}$_empty	$P(F\ (empty \wedge jobs = 0) \wedge G\neg(x \neq 1 \wedge y \neq 2))$	0.0
	pass_empty	$P(F\ (empty \wedge pass))$	0.0
	$\overline{\text{pass}}$_empty	$P(F\ (empty \wedge \neg pass))$	0.0
Coll.	collision	$P(F_{\leq 100}\ collision)$	0.1
SG	blackout	$P(F_{\leq 100}\ blackout)$	0.2
SM	bankruptcy	$P(F\ bankruptcy)$	0.0

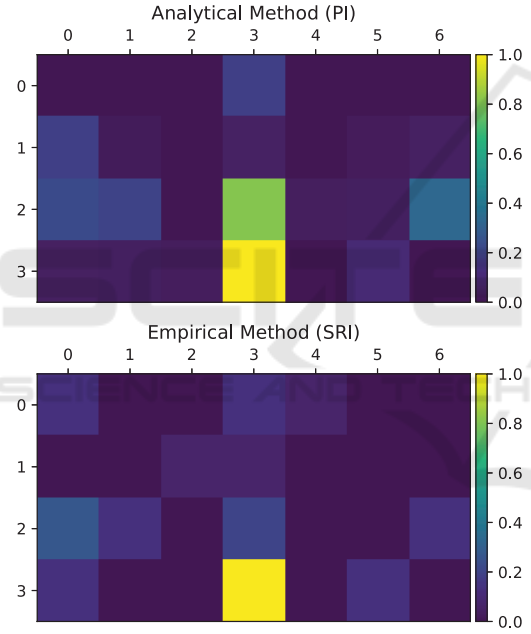


Figure 3: Freeway feature impacts (normalized between 0 and 1) for the PI and SRI method.

Does the PI method have the same behavior as the related SRI method? We compare the results of our PI approach to the empirical SRI approach (Chan et al., 2020) in the Freeway environment using the reward function and the expected reachability probability of crossing the street (see Figure 3). We generate both the SRI and PI maps using a sample size of $N = 300$ and an $\varepsilon = 1$. The results show that both approaches yield similar results.

Can the PI method generate different property impacts for different advanced property queries? We now show that PI is suited to measure the property impact for properties that can not be expressed by rewards which we call here *advanced property queries* (see Figure 4). To make the interpretation of advanced properties more straightforward, we focus on the Taxi environment and use the advanced property queries *deadlock1* and *station_empty*. Advanced property queries contain, for example, the U-operator (Definition 3.3), which allows the adversary to make sure that certain events happen before other events. Figure 4 shows the property impact of each attack on the policy and different ε-bounded attacks. By attacking the *done* feature via an PIA (with $\varepsilon = 1$), it is possible to drive the taxi around without running out of fuel and not finishing jobs while having a passenger on board (deadlock1). Figure 4 also shows that it is possible to let the taxi drive first to the gas station and let it run out of fuel afterwards (station_empty). We observe that for different ε-bounds, PIAs have different impacts via features on the temporal logic properties (see station_empty in Figure 4).

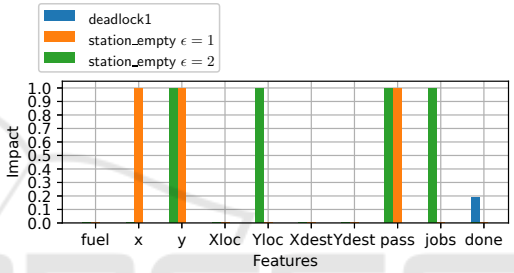


Figure 4: Taxi environment. This diagram plots different advanced property impacts of different PIAs. The original property values (without an attack) are all zero.

What are the limitations of PIAs? We now analyze the limitations of PIAs and compare them with the FGSM attack (baseline) and the robustness checker. For each experiment, we ε-bounded all the generated attacks for a fair comparison.

Table 2: Impact* stands for the optimal adversarial attack impact ($|P^{max} - P|$) via the feature specified in *Features*, P^{max} for the maximal probability $P^{max}(\phi)$ with an attack, P for the original probability $P(\phi)$ (without an attack), Time in seconds, C for *Collision Avoidance*, SG for *Smart Grid*, SM for *Stock Market*, Baseline is a standard FGSM attack on the whole observation.

Setup				**Robustness Checker**				**PIA**		**Baseline (FGSM)**	
Env.	*Features*	ε	*Property Query*	P^{max}	P	*Impact**	*Time*	*Impact*	*Time*	*Impact*	*Time*
Taxi	done	1	deadlock1	0.44	0.0	0.44	9	0.19	20	0.00	6
	done	1	deadlock2	0.00	0.0	0.00	9	0.00	20	0.00	6
	fuel	2	pass_empty	1.00	0.0	1.00	25	0.25	20	0.00	6
	y	2	$\overline{\text{pass}}$_empty	1.00	0.0	1.00	27	1.00	20	1.00	6
	x	1	station_empty	1.00	0.0	1.00	24	1.00	6	1.00	6
	x	1	$\overline{\text{station}}$_empty	1.00	0.0	1.00	30	1.00	6	1.00	6
C	obs1_x	1	collision	0.87	0.1	0.86	65	0.46	213	0.87	211
SG	non_renewable	1	blackout	0.97	0.2	0.95	2	0.39	2	0.98	2
SM	sell_price	1	bankruptcy	0.81	0.0	0.81	15	0.08	20	0.00	4

Table 2 shows that PIAs, in comparison to FGSM attacks, have similar impacts on temporal logic properties (compare *impact* columns of PIA and FGSM). For temporal logic properties where some correct decision-making is still needed, PIAs perform better than the FGSM attack (for instance, *pass_empty*). However, PIAs do not necessarily create a maximal impact on the property values like the robustness checker method (compare *PIA impact* with *Impact**).

After observing the results of the three methods (PIA, FGSM, robustness checker), we can summarize. By verifying the robustness of the trained RL policies, the adversary can already extract for each state the optimal adversarial attack that is part of the most impactful attack. Since PIAs build induced DTMCs and the robustness checker induced MDPs, PIAs are suited for MDPs with more states and transitions before running out of memory (see Gross et al., 2022a, for more details about the limitations of model checking RL policies).

Does adversarial training make trained RL policies more robust against PIAs? Figure 4 shows that an adversarial attack (bounded by $\varepsilon = 1$) on feature *done* can bring the taxi agent into a deadlock and lets it drive around after the first job is done ($deadlock1 = 0.19$). To protect the RL agent from this attack, we trained the RL taxi policy over 5000 additional episodes via adversarial training by using our method PIA on the done feature to make the policy more robust against this deadlock attack. The adversarial training improves the feature robustness for the done feature (0) but deteriorates the robustness for the other features (all other feature PI-values: 1). That agrees with the observation that adversarially trained RL policies may be less robust to other types of adversarial attacks (Zhang et al., 2020; Korkmaz, 2021a, 2022).

ACKNOWLEDGEMENTS

This research has been funded by the Dutch NWO grant NWA.1160.18.238 (PrimaVera); the Flemish interuniversity iBOF "DESCARTES" and FWO "SAILor" projects (G030020N).

REFERENCES

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Baier, C. and Katoen, J. (2008). *Principles of model checking*. MIT Press.

Carlini, N. and Wagner, D. A. (2017). Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57. IEEE Computer Society.

Cassez, F., David, A., Fleury, E., Larsen, K. G., and Lime, D. (2005). Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR*, pages 66–80. Springer.

Chan, P. P. K., Wang, Y., and Yeung, D. S. (2020). Adversarial attack against deep reinforcement learning with static reward impact map. In *AsiaCCS*, pages 334–343. ACM.

Clark, G. W., Doran, M. V., and Glisson, W. (2018). A malicious attack on the machine learning policy of a robotic system. In *TrustCom/BigDataSE*, pages 516–521. IEEE.

Courcoubetis, C. and Yannakakis, M. (1988). Verifying temporal properties of finite-state probabilistic programs. In *FOCS*, pages 338–345. IEEE Computer Society.

Courcoubetis, C. and Yannakakis, M. (1995). The complexity of probabilistic verification. *J. ACM*, 42(4):857–907.

David, A., Jensen, P. G., Larsen, K. G., Mikucionis, M., and Taankvist, J. H. (2015). Uppaal Stratego. In *TACAS*, pages 206–211. Springer.

Dräger, K., Forejt, V., Kwiatkowska, M. Z., Parker, D., and Ujma, M. (2015). Permissive controller synthesis for probabilistic systems. *Log. Methods Comput. Sci.*, 11(2).

Farazi, N. P., Zou, B., Ahamed, T., and Barua, L. (2021). Deep reinforcement learning in transportation research: A review. *Transportation Research Interdisciplinary Perspectives*, 11:100425.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. (2020). Adversarial policies: Attacking deep reinforcement learning. In *ICLR*. OpenReview.net.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *ICLR*.

Gross, D., Jansen, N., Junges, S., and Pérez, G. A. (2022a). COOL-MC: A comprehensive tool for reinforcement learning and model checking. In *SETTA*. Springer.

Gross, D., Simão, T. D., Jansen, N., and Perez, G. A. (2022b). Targeted adversarial attacks on deep reinforcement learning policies via model checking. *CoRR, abs/2212.05337*.

Hahn, E. M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., and Wojtczak, D. (2019). Omega-regular objectives in model-free reinforcement learning. In *TACAS (1)*, pages 395–412. Springer.

Hansson, H. and Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects Comput.*, 6(5):512–535.

Hasanbeig, M., Kroening, D., and Abate, A. (2020). Deep reinforcement learning with temporal logics. In *FORMATS*, pages 1–22. Springer.

Hensel, C., Junges, S., Katoen, J., Quatmann, T., and Volk, M. (2022). The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610.

Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. (2017). Adversarial attacks on neural network policies. In *ICLR*. OpenReview.net.

Ilahi, I., Usama, M., Qadir, J., Janjua, M. U., Al-Fuqaha, A. I., Hoang, D. T., and Niyato, D. (2022). Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *IEEE Trans. Artif. Intell.*, 3(2):90–109.

Korkmaz, E. (2021a). Adversarial training blocks generalization in neural policies. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

Korkmaz, E. (2021b). Investigating vulnerabilities of deep neural policies. In *UAI*, pages 1661–1670. AUAI Press.

Korkmaz, E. (2022). Deep reinforcement learning policies learn shared adversarial features across mdps. In *AAAI*, pages 7229–7238. AAAI Press.

Kwiatkowska, M. Z., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591. Springer.

Lee, X. Y., Esfandiari, Y., Tan, K. L., and Sarkar, S. (2021). Query-based targeted action-space adversarial policies on deep reinforcement learning agents. In *ICCPS*, pages 87–97. ACM.

Lee, X. Y., Ghadai, S., Tan, K. L., Hegde, C., and Sarkar, S. (2020). Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In *AAAI*, pages 4577–4584. AAAI Press.

Lin, Y., Hong, Z., Liao, Y., Shih, M., Liu, M., and Sun, M. (2017a). Tactics of adversarial attack on deep reinforcement learning agents. In *ICLR*. OpenReview.net.

Lin, Y., Liu, M., Sun, M., and Huang, J. (2017b). Detecting adversarial attacks on neural network policies with visual foresight. *CoRR*, abs/1710.00814.

Liu, Z., Guo, Z., Cen, Z., Zhang, H., Tan, J., Li, B., and Zhao, D. (2022). On the robustness of safe reinforcement learning under observational perturbations. *CoRR*, abs/2205.14691.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533.

Nakabi, T. A. and Toivanen, P. (2021). Deep reinforcement learning for energy management in a microgrid with flexible demand. *Sustainable Energy, Grids and Networks*, 25:100413.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. (2017). Robust adversarial reinforcement learning. In *ICML*, pages 2817–2826. PMLR.

Rakhsha, A., Radanovic, G., Devidze, R., Zhu, X., and Singla, A. (2020). Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *ICML*, pages 7974–7984. PMLR.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Vamplew, P., Smith, B. J., Källström, J., de Oliveira Ramos, G., Radulescu, R., Roijers, D. M., Hayes, C. F., Heintz, F., Mannion, P., Libin, P. J. K., Dazeley, R., and Foale, C. (2022). Scalar reward is not enough: a response to silver, singh, precup and sutton (2021). *Auton. Agents Multi Agent Syst.*, 36(2):41.

Yu, M. and Sun, S. (2022). Natural black-box adversarial examples against deep reinforcement learning. In *AAAI*, pages 8936–8944. AAAI Press.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D. S., and Hsieh, C. (2020). Robust deep reinforcement learning against adversarial perturbations on state observations. In *NeurIPS*, pages 21024–21037.