



HAL
open science

Robust Deep Reinforcement Learning Through Adversarial Attacks and Training: A Survey

Lucas Schott, Josephine Delas, Hatem Hajri, Elies Gherbi, Reda Yaich, Nora
Boulahia-Cuppens, Frederic Cuppens, Sylvain Lamprier

► **To cite this version:**

Lucas Schott, Josephine Delas, Hatem Hajri, Elies Gherbi, Reda Yaich, et al.. Robust Deep Reinforcement Learning Through Adversarial Attacks and Training: A Survey. 2024. hal-04521876

HAL Id: hal-04521876

<https://hal.science/hal-04521876>

Preprint submitted on 26 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Robust Deep Reinforcement Learning Through Adversarial Attacks and Training : A Survey

Lucas Schott^{1,2†}, Joséphine Delas^{1,3†}, Hatem Hajri^{1,4},
Elies Gherbi¹, Reda Yaich¹, Nora Boulahia-Cuppens³,
Frederic Cuppens³, Sylvain Lamprier^{2,5}

¹IRT SystemX, Palaiseau, 91120, France.

²MLIA, ISIR, Sorbonne Université, Paris, 75005, France.

³Polytechnique Montréal, Montréal, 6079, Québec, Canada.

⁴Safran Tech, Châteaufort, 78117, France.

⁵LERIA, Université d'Angers, Angers, 49000, France.

†These authors contributed equally to this work.

Abstract

Deep Reinforcement Learning (DRL) is an approach for training autonomous agents across various complex environments. Despite its significant performance in well known environments, it remains susceptible to minor conditions variations, raising concerns about its reliability in real-world applications. To improve usability, DRL must demonstrate trustworthiness and robustness. A way to improve robustness of DRL to unknown changes in the conditions is through Adversarial Training, by training the agent against well suited adversarial attacks on the dynamics of the environment. Addressing this critical issue, our work presents an in-depth analysis of contemporary adversarial attack methodologies, systematically categorizing them and comparing their objectives and operational mechanisms. This classification offers a detailed insight into how adversarial attacks effectively act for evaluating the resilience of DRL agents, thereby paving the way for enhancing their robustness.

Keywords: Deep Reinforcement Learning, Robustness, Adversarial Attacks, Adversarial Training

1 Introduction

The advent of Deep Reinforcement Learning (DRL) has marked a significant shift in various fields, including games [1–3], autonomous robotics [4], autonomous driving [5], and energy management [6]. By integrating Reinforcement Learning (RL) with Deep Neural Networks (DNN), DRL can leverage high dimensional continuous observations and rewards to train neural policies, without the need for supervised example trajectories.

While DRL achieves remarkable performances in well known controlled environments, it also encounters challenges in ensuring robust performance amid diverse condition changes and real-world perturbations. It particularly struggles to bridge the reality gap [7, 8], often DRL agents are trained in simulation that remains an imitation of the real-world, resulting in a gap between the performance of a trained agent in the simulation and its performance once transferred to the real-world application. Even without trying to bridge the reality gap, agents can be trained in the first place in some conditions, and be deployed later and the conditions may have changed since. This poses the problem of robustness, which refers to the agent’s ability to maintain performance in deployment despite slight condition changes in the environment or minor perturbations.

Moreover the emergence of adversarial attacks that generate perturbation in the inputs and disturbances in the dynamics of the environment, which are deliberately designed to mislead neural network decisions, poses unique challenges in RL [9, 10] and can be a key to resolve the robustness problem in RL, necessitating further exploration and understanding.

This survey aims to address these critical areas of concern. It focuses on key issues by presenting a comprehensive framework for understanding the concept of robustness of DRL agent. It covers both robustness to perturbed inputs as well as robustness to perturbed dynamics of the environment. Additionally, it introduces a new classification system that organizes every type of perturbation affecting robustness into a unified model. It also offers a review of the existing literature on adversarial methods for robust DRL agents and classifies the existing methods in the proposed taxonomy. The goal is to provide a deeper understanding of various adversarial techniques, including their strengths, limitations, and the impact they have on the performance, robustness and generalization capabilities of DRL agents.

Historically, the primary focus on adversarial examples has been in the realm of supervised learning [11]. Attempts to extend this scope to RL have been made, but these have primarily concentrated on adversarial evasion methods and robustness-oriented classification [9, 12]. To bridge this gap, our work introduces a robustness-centric study of adversarial methods in DRL.

The key contributions of this work include:

- Formalizing the concept of Robustness in DRL.
- Developing a taxonomy and classification for adversarial attack in DRL.
- Reviewing existing adversarial attack, characterized using our proposed taxonomy.
- Reviewing how adversarial attacks can be used to improve robustness of DRL agents.

The structure of the survey is organized as follows: Section 2 provides an introduction to RL and the security implications of DNNs, as well as the mathematical prerequisites for analyzing RL Robustness. Section 3 introduces a formalization of the notion of Robustness in DRL. Section 4 presents a taxonomy for categorizing adversarial attack methods as shown in Figure 1. Sections 5.1 and 5.2 explore observation and dynamic alterations attacks, respectively. Finally section 6 focuses on strategies for applying adversarial attacks and adversarial training.

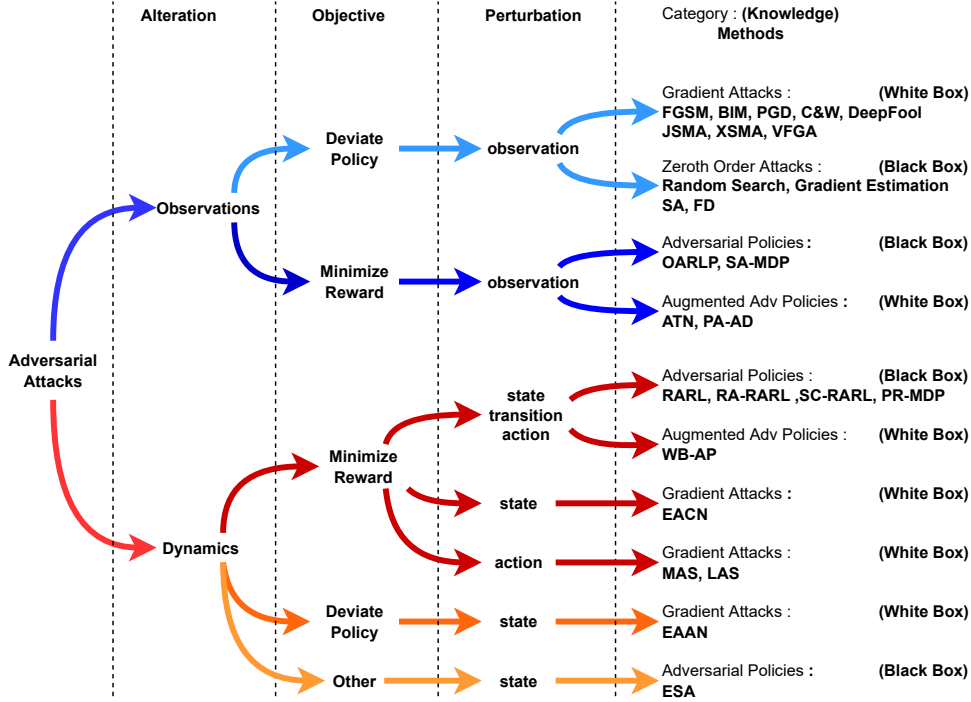


Fig. 1: Categorization of the adversarial attacks of the literature as described in Section 5 with the taxonomy introduced Section 4 of this survey.

2 Background

2.1 Reinforcement Learning

RL focuses on decision-making in dynamic environments [13]. RL agents learn by interacting with an environment: they take actions and receive feedback in terms of numerical rewards. The objective of a RL agent is to learn a policy, a mapping from states to actions, which maximizes the expected cumulative reward over time.

2.1.1 Partially Observable Markov Decision Process

A Markov Decision Process (MDP) is a mathematical framework for modeling decision-making problems where an agent interacts with an environment over discrete time steps. In most real-world applications, the agent may not have access to the environment’s complete states and instead receives partial observations. This scenario is known as a Partially Observable Markov Decision Process (POMDP), which is a generalization of the MDP framework, represented by the tuple $\Omega = (S, A, T, R, X, O)$, where:

- S is the set of states in the environment,
- A is the set of actions available to the agent,
- $T : S \times S \times A \rightarrow [0, 1]$ is the stochastic transition function, with $T(s_+|s, a)$ denoting the probability of transitioning to state s_+ given state s and action a ,
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function. $R(s, a, s_+)$ is received by the agent for taking action a in state s and moving to state s_+ ,
- X is the set of observations as perceived by the agent,
- $O : S \times X \rightarrow [0, 1]$ is the observation function, with $O(x|s)$ denoting the probability of observing x given state s .

A step in the environment represented by the POMDP Ω is represented by the transition (s_t, x_t, a_t, s_{t+1}) , where s_t stands for the state, x_t the observation of this state, a_t the action applied by the agent, s_{t+1} the next state after transition. In this paper, we will use the POMDP framework as a general model, even though some environments could be described as MDPs.

2.1.2 Fundamentals of Reinforcement Learning

In RL, the goal is to learn a policy $\pi : A \times S \rightarrow [0, 1]$, $\pi(a|s)$ denoting the probability of selecting the action a given state s . The optimal policy, denoted as π^* , therefore maximizes the expected cumulative discounted reward :

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi^{\Omega}} [R(\tau)]$$

with

$$R(\tau) = \sum_{t=0}^{|\tau|} \gamma^t R(s_t, a_t, s_{t+1})$$

where $\tau = (s_0, a_0, s_1, \dots, s_{|\tau|})$ is sampled from the distribution π^{Ω} of trajectories obtained by executing policy π in environment Ω . The discount factor γ , ranging from 0 to 1, weights the importance of future rewards.

An important criterion for defining optimality is the state value function, denoted as $V^{\pi} : S \rightarrow \mathbb{R}$. For a state s , the value $V^{\pi}(s)$ represents the expected cumulative discounted reward starting from s and following the policy π thereafter. This can be formally expressed as:

$$V^{\pi}(s) = \mathbb{E}_{\tau \sim \pi^{\Omega}} [R(\tau) | s_0 = s] \tag{1}$$

It can be expressed recursively with the Bellman equation :

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s_+} T(s_+|s, a) \left(R(s, a, s_+) + \gamma V^\pi(s_+) \right)$$

Finally, the state-action value function $Q^\pi : S \times A \rightarrow \mathbb{R}$ is used in many algorithms as an alternative to V^π . The Q-value function of a state s and action a is the expected cumulative discounted reward, starting from s , taking a , and following π :

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi^\Omega} [R(\tau) | s_0 = s, a_0 = a] \quad (2)$$

It can be expressed recursively with the equation :

$$Q^\pi(s, a) = \sum_{s_+} T(s_+|s, a) \left(R(s, a, s_+) + \gamma \sum_{a_+} \pi(a_+|s_+) [Q^\pi(s_+, a_+)] \right)$$

In the POMDP setting, since states are not directly observable by agents, the practice is to base policies and value functions on the history of observations (i.e., $x_{0:t}$ at step t) in place of the true state of the system (i.e., s_t). For the ease of notations, we consider in the following policies and value functions defined with only the last observation as input (i.e., x_t), while every approach presented below can be extended to methods leveraging full histories of observations. More specifically, we consider in the following policies defined as $\pi : A \times X \rightarrow [0; 1]$ and action-value functions as $Q : A \times X \rightarrow \mathbb{R}$. Figure 2 shows the flowchart of an agent with a policy function π interacting with a POMDP environment.

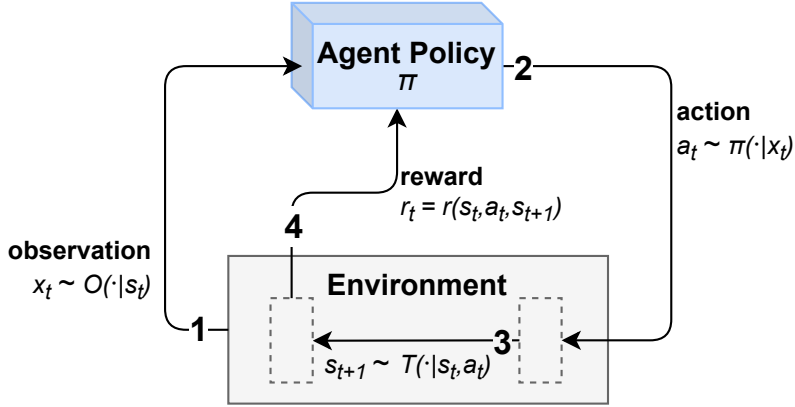


Fig. 2: Flowchart of an agent with a policy function π interacting with a POMDP environment

2.2 Neural Networks and Deep Reinforcement Learning

To solve the complex task of RL problems in a large input space and enable generalization, RL methods are combined with DNNs.

2.2.1 Deep Neural Networks (DNNs)

A neural network is a system of interconnected nodes (neurons) that process and transmit signals. DNNs are models utilizing multiple layers of neurons, featuring varying degrees of architecture complexity, to analyze intricate data patterns. Training involves adjusting inter-neuron weights parameters to reduce errors (called loss function) between the network’s predictions and actual outcomes, often employing Stochastic Gradient Descent (SGD) inspired algorithms. This training refines the network’s ability to recognize and respond to input data accurately. The update rule of the parameters θ of the model f_θ in this context, given inputs x , labels y , learning rate α and loss function \mathcal{L} , is expressed as:

$$\theta = \theta - \alpha \cdot \nabla_{\theta} \mathcal{L}(f_{\theta}(x), y)$$

2.2.2 Deep Reinforcement Learning (DRL)

DRL combines the principles of RL with the capabilities of DNNs. The central concept in DRL is to construct a policy π using a DNN. This can be achieved either by approximating the Q-function (as in Equation (2)), the V-function (as in Equation (1)), or by directly inferring the policy from experiences. There are several popular DRL algorithms, each with their specific strengths and weaknesses, some are better suited for specific context like discrete or continuous action space, or depending on possibility to train the DNNs on- or off-policy. The fundamental DRL algorithms are PG [14], DQN [15] and DDPG [16], but the most effective contemporary algorithms are Rainbow [17], PPO [18], SAC [19] or TQC [20] depending on the context.

2.3 Security challenges in DNNs

DNNs are a powerful tool now used in numerous real-world applications. However, their complex and highly non-linear structure make them hard to control, raising growing concerns about their reliabilities. Adv ML recently emerged to exhibit vulnerabilities of DNNs by processing attacks on their inputs that modify outcomes. NIST’s National Cybersecurity Center of Excellence (NCCE) [21] and its European counterpart, ETSI Standards [22], provide terminologies and ontologies to frame the study of these adversarial methods.

2.3.1 Adversarial Machine Learning

Adversarial attacks are initially designed to exploit vulnerabilities in DNNs, threatening their privacy, availability, or integrity [23]. If the term adversarial attack suggests a malicious intention, it is to be noted that these methods may also be used by a model’s owner to improve its performances and assess its vulnerability. Indeed, adversarial ML aims to analyze the capabilities of potential attackers, comprehend the

impact of their attacks and develop ML algorithms able to withstand these security threats. An adversary may act during the learning phase by poisoning the training data, or the inference phase modifying the inputs to evade decision. In this paper we consider robustness of already trained models, as well as leveraging adversarial examples as a defense method during the training phase in order to improve robustness at the inference phase, therefore we focus on discussing model robustness to evasion methods.

2.3.2 Adversarial Examples [21]

The large number of dimensions of a DNNs input space inevitably leads to blind spots and high sensitivity to small perturbations. In the restrained domain of classification, Adversarial examples are slightly altered data instances, carefully crafted to trick the model into misclassification while staying undetected. Computation techniques range from costly hand-made modifications [24] to perturbations generated by complex algorithms, yet the fundamental objective of adversarial example generation remains simple and can be summarized in Equation (3): given the original instance x , find the closest example x' relative to the chosen metric $\|\cdot\|$ that leads a model's function f_θ to change its output.

$$\min_{x'} \|x - x'\| \quad s.t. \quad f_\theta(x) \neq f_\theta(x') \quad (3)$$

A variety of perturbation methods exist for the supervised classification problem, depending on the adversary's objective and the model's constraints. A extensive overview of these methods can be found together with defense strategies in [11].

2.4 Security challenges in DRL

DRL enables agents to learn complex behaviors by interacting with their environment. However, this interaction introduces unique security challenges not fully encountered in traditional deep learning contexts. The dynamic nature of DRL, combined with the necessity for long-term strategic decision-making, exposes DRL systems to a range of security threats that can compromise their learning process, decision-making integrity, and overall effectiveness. These challenges are further exacerbated by the adversarial landscape, where attackers can manipulate the environment or the agent's perception to induce faulty learning or decision-making. Addressing these challenges is crucial for deploying DRL in security-sensitive applications.

2.4.1 Safe RL Control

Formulating the challenge of safe control in RL [25] merges insights from the realms of control theory and reinforcement learning, aiming to optimize a solution that balances task achievement with stringent safety standards. At the heart of this approach lies three critical elements: the dynamic system behavior encapsulated within a model of the agent, the objectives or targets of the control task expressed through a cost function, and a set of safety constraints that the solution must adhere to. The goal is to develop a policy or controller that is capable of producing the necessary actions

to navigate the system towards its objectives, all while strictly complying with the predefined safety protocols.

2.4.2 Reality Gap, Real World Perturbations and Generalization

The Reality Gap [7, 8] refers to the divergence between the simulated training environments of DRL agents and the complex, unpredictable conditions they encounter in real-world applications. This discrepancy challenges not only the agent’s ability to generalize across different contexts but also presents a profound security vulnerability. Real-world perturbations—unexpected changes in the environment—can lead to degraded performance or entirely erroneous actions by the DRL agent, particularly when these agents are confronted with scenarios slightly different from their training conditions. Such perturbations may arise naturally, from unmodeled aspects of the environment, or be adversarially crafted, with the intent to exploit these generalization weaknesses and induce failures. Addressing the Reality Gap, thereby enhancing the agents’ ability to generalize effectively and securing them against both natural and adversarial perturbations, is crucial for the safe and reliable deployment of DRL systems in environments demanding high levels of security and robust decision-making.

2.4.3 Robust RL Control

Robust RL control introduces an advanced framework for RL by incorporating elements of uncertainty, such as parametric variations and external disturbances, into the system dynamics [26]. This approach shifts the optimization focus towards minimizing the maximum possible loss, essentially preparing the system to handle the worst-case scenario efficiently. It does so through a min-max optimization strategy, where the goal is to find a control policy that minimizes the maximum expected cost.

$$\min_{\pi} \max_{\delta \in \Delta} J(\pi, \delta)$$

Where $J(\pi, \delta)$ represents the expected cost (or loss) of policy π when subjected to perturbations δ introduced by the adversary. The set Δ defines the allowable perturbations or disturbances.

This methodology ensures that the control system remains effective and reliable even when faced with unpredictable changes or adverse conditions, thereby enhancing its robustness and resilience in uncertain environments. This framework for enhancing robust control in RL, can participate for generalization of the policies across conditions changes thus helping to bridge the reality gap and overcome real world perturbations.

2.4.4 Adversarial Attacks of DRL

If adversarial attacks were historically developed for supervised image classification models, they were proven equally effective for DRL agents. [27] first established the vulnerability of DQNs to adversarial perturbations, their statement soon supported by further studies [23]. Moreover, the RL framework offers more adversarial possibilities than the simple adaptation of supervised methods. Indeed, various components of the POMDP can be found vulnerable (like observation or transition function) through

various elements which could be critic entry points (observations, states, actions), while the long-term dependencies inherent to DRL raise complex security challenges. On the other hand, this higher level of adversarial latitude enables new defense strategies for improving the agents robustness. This survey explores how adversarial attacks in RL can be used to generate perturbations that result in the worst-case scenarios crucial for Robust RL Control.

3 Formalization and Scope

The existence of adversarial examples poses a significant threat for DRL agents, particularly in applications where incorrect predictions can have serious consequences, such as autonomous driving or medical diagnosis. Developing robust DRL algorithms that can defend against adversarial attacks and bridge the reality gap is an important and active research area in the field.

This survey aims to identify and assess how using adversarial examples during policy training can improve agent robustness. More specifically, we discuss the ability of various types of adversarial generation strategies to help anticipate the reality gap, which refers to a discrepancy between the training environment (e.g., a simulator) and the deployment one (which can include perturbations, whether they can be adversarially generated or not).

3.1 The problem of Robustness in RL

Generally speaking, we are interested in the following optimization problem:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\Omega \sim \Phi(\Omega|\pi)} \mathbb{E}_{\tau \sim \pi^{\Omega}(\tau)} [R(\tau)]$$

where Φ corresponds to the distribution of environments to which the agent is likely to be confronted when deployed (whether it adversarially considers π or not at test time), $\pi^{\Omega}(\tau)$ is the distribution of trajectories using the policy π and the dynamics from Ω , and $R(\tau)$ is the cumulative reward collected in τ . While this formulation suggests meta-reinforcement learning, in our setting $\Phi(\Omega|\pi)$ is unknown at train time. The training setup is composed of a unique MDP on which the policy can be learned, which is usually the case for many applications.

Given a unique training POMDP Ω , the problem of robustness we are interested in can be reformulated by means of an alteration distribution $\Phi(\phi|\pi)$:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\phi \sim \Phi(\phi|\pi)} \mathbb{E}_{\tau \sim \pi^{\phi, \Omega}(\tau)} [R(\tau)]$$

where $\pi^{\phi, \Omega}$ is the distribution of trajectories using policy π on $\phi(\Omega)$, standing as the MDP Ω altered by ϕ . Generally speaking, we can set ϕ as a function that can alter any component of Ω as $\phi(\Omega) = (\phi_S(S^{\Omega}), \phi_A(A^{\Omega}), \phi_T(T^{\Omega}), \phi_R(R^{\Omega}), \phi_X(X^{\Omega}), \phi_O(O^{\Omega}))$. As discussed below and also in [28], while ϕ can simultaneously affect any of these components, we particularly focus on two crucial components for robustness:

- Observation alterations: ϕ_O denotes alterations of the observation function of Ω . In the corresponding altered environment $\tilde{\Omega} = (S^\Omega, A^\Omega, T^\Omega, R^\Omega, X^\Omega, \phi_O(O^\Omega))$, the observation obtained from a state $s \in S^\Omega$ could differ from that in Ω . This can result from an adversarial attacker, that perturb signals from sensors to induce failures, observation abilities from real world that might be different than in simulation, or even unexpected failures of some sensors. These perturbations only induce perception alterations for π , without any effect on the true internal state of the system in the environment. Occurring at a specific step t of a trajectory τ , such alteration thus only impacts the future of τ if it induces changes in the policy decision at t .
- Dynamics alterations: ϕ_T denotes alterations of the transition function of Ω . In the corresponding altered environment $\tilde{\Omega} = (S^\Omega, A^\Omega, \phi_T(T^\Omega), R^\Omega, X^\Omega, O^\Omega)$, dynamics are modified, such that actions have not the exactly same effect as in Ω . This can result from an adversarial attacker, that modifies components of the environment to induce failures, from real world physics, that might be different than those from the training simulator, or from external events, that can incur unexpected situations. Dynamics alterations act on trajectories by modifying the resulting state s_{t+1} emitted by the transition function T at any step t . Even when localized at a single specific step t of a trajectory, they thus impact its whole future.

In this work, we do not explicitly address variation of other components (S , A , R and X), as they usually pertain to different problem areas. ϕ_S (resp. ϕ_A) denotes alterations of the state (resp. action) set, where states (resp. actions) can be removed or introduced in S^Ω (resp. A^Ω). ϕ_X denotes alterations of the observation support X^Ω . While some perturbations of dynamics ϕ_T or observations ϕ_O can lead the agent to reach new states or observations never considered during training (which corresponds to implicit ϕ_S or ϕ_X perturbations), ϕ_S , ϕ_A , and ϕ_X all correspond to support shifts, related to static Out-Of-Domain issues, which we do not specifically focus on in this work. ϕ_R denotes alterations of the reward function, which does not pose problem of robustness in usage, since the reward function is only used during training.

3.2 Adversarial Attacks for Robust RL

Following distributionally robust optimization (DRO) principles [29], unknown distribution shifts can be anticipated by considering worst-case settings in some uncertainty sets \mathcal{R} . In our robust RL setting, this comes down to the following max-min optimization problem:

$$\pi^* = \arg \max_{\pi} \min_{\tilde{\Phi} \in \mathcal{R}} \mathbb{E}_{\phi \sim \tilde{\Phi}(\phi|\pi)} \mathbb{E}_{\tau \sim \pi^{\phi, \Omega}(\tau)} [R(\tau)] \quad (4)$$

where \mathcal{R} is a set of perturbation distributions. As well-known in DRO literature for supervised regression problems, the shape of \mathcal{R} has a strong impact on the corresponding optimal decision system. In our RL setting, increasing the level of disparities allowed by the set \mathcal{R} constrains the resulting policy π to have to perform simultaneously over a broader spectrum of environmental conditions. While this enables better

generalization for environmental shifts, it also implies to deal with various highly unrealistic scenarios if the set \mathcal{R} is not restricted on reasonable levels of perturbations. With extremely large sets \mathcal{R} , the policy π is expected to be equally effective for any possible environment, eventually converging to a trivial uniform policy, that allocates equal probability to every action for any state from S^Ω . The shape of \mathcal{R} has thus to be controlled to find an accurate trade-off between generalization and effectiveness. This is done in the following by setting restricted supports of perturbation.

In our setting, dealing with worst-case distributions of perturbations defined over full supports of Ω is highly intractable in most realistic applications. In this survey, we rather focus on adversarial training that leverage the simultaneous optimization of an attacker agent ξ , that produces perturbations for situations reached by the protagonist π , by acting on adversarial actions A_ξ^Ω that the environment Ω permits:

$$\begin{aligned} \pi^* &= \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi^{\xi^*, \Omega}(\tau)} [R(\tau)] \\ \text{s.t.} \quad \xi^* &= \arg \min_{\xi} \Delta^{\pi, \Omega}(\xi) \end{aligned}$$

where $\Delta^{\pi, \Omega}(\xi)$ stands as the optimization objective of the adversarial agent given π and the training environment Ω , which ranges from adverse reward functions to divergence metrics (c.f., section 4.3), and $\pi^{\xi, \Omega}(\tau)$ corresponds to the probability of a trajectory following policy π in a POMDP dynamically modified by an adversarial agent ξ , given a set of actions $A_\xi^\Omega = (A_{\xi, X}^\Omega, A_{\xi, A}^\Omega, A_{\xi, S}^\Omega, A_{\xi, T}^\Omega, A_{\xi, S+}^\Omega)$. The action $a_t^\xi = (a_t^{\xi, X}, a_t^{\xi, A}, a_t^{\xi, S}, a_t^{\xi, T}, a_t^{\xi, S+})$ of adversary ξ can target any element of any transition $\tau_t = (s_t, x_t, a_t, s_{t+1})$ of trajectories in Ω . While any perturbation of x_t induce an alteration of the observation function O^Ω , any perturbation of s_t , a_t or s_{t+1} induce an alteration of the transition function T^Ω (either directly, through its internal dynamics, or indirectly via the modification of its inputs or outputs).

In this setting, any trajectory is composed as a sequence of adversary-augmented transitions $\tilde{\tau}_t = (s_t, x_t, a_t, a_t^\xi, x'_t, a'_t, \tilde{s}_t, \tilde{x}_t, \tilde{s}_{t+1}, \tilde{x}_{t+1}, s_{t+1})$, where the elements x'_t (resp. a'_t) stands for the perturbed observation (resp. action) produced by the application of the adversary action $a_t^{\xi, X}$ (resp. $a_t^{\xi, A}$) at step t . \tilde{s}_t (resp. \tilde{s}_{t+1}) stands for the intermediary state produced by the application of the adversary action $a_t^{\xi, S}$ (resp. $a_t^{\xi, T}$) at step t before (resp. during) the transition function, and \tilde{x}_t (resp. \tilde{x}_{t+1}) is the observation of this state. Finally s_{t+1} stands for the final next state produced by the application of the adversary action $a_t^{\xi, S+}$ after the transition function, its observation is x_{t+1} . The support and scope of adversarial actions define the level of perturbations allowed in the corresponding uncertainty set \mathcal{R} from (4), with impacts on the generalization/accuracy trade-off of the resulting policy π . While the protagonist agent π acts from x_t with $a_t \sim \pi(\cdot | x_t)$, in the following, we consider the general case of adversaries ξ that act from s_t , x_t and a_t , that is $\xi : S^\Omega \times X^\Omega \times A^\Omega \times A_\xi^\Omega \rightarrow [0; 1]$ where $a_t^\xi \sim \xi(\cdot | s_t, x_t, a_t)$. By doing this we consider adversaries ξ that have full knowledge of the environment, observation and action, while this could be easily limited to adversarial policies ξ that act only from partial information.

4 Taxonomy of Adversarial Attacks of DRL

We conduct a systematic analysis of adversarial attacks for RL agents, with a focus on their purposes and applications. To better grasp the variety of methods available and their specificities, we propose a taxonomy of adversarial attacks for DRL. This taxonomy is used to categorize the adversarial attack as previously shown in Figure 1 and later described in Table 1. This section discusses the different components of adversarial approaches for robust RL, before developing main approaches in the next section.

4.1 Perturbed Element

An adversarial attacks is a method that use an adversarial action $a_t^\xi \in A_\xi^\Omega$ emitted by the adversary agent ξ at step t , to produce a perturbation in the simulation during the trajectory of an agent. Given the type of attack, an action a_t^ξ can directly perturb different elements :

The observations x_t

Via a perturbation function $O^\xi : X^\Omega \times X^\Omega \times A_{\xi,X}^\Omega \rightarrow [0;1]$, where $x'_t \sim O^\xi(\cdot|x_t, a_t^{\xi,X})$.

The actions a_t

Via a perturbation function $A^\xi : A^\Omega \times A^\Omega \times A_{\xi,A}^\Omega \rightarrow [0;1]$ where $a'_t \sim A^\xi(\cdot|a_t, a_t^{\xi,A})$.

The current state s_t (before transition)

Via an additional transition functions $T_{\xi-}^\Omega : S^\Omega \times S^\Omega \times A_{\xi,S}^\Omega \rightarrow [0;1]$ where $\tilde{s}_t \sim T_{\xi-}^\Omega(\cdot|s_t, a_t^{\xi,S})$ is applied after the decision a_t of the agent is taken and before the main transition function of the environment is applied.

The transition function T^Ω

Via an adversarially augmented transition functions $T_\xi^\Omega : S^\Omega \times S^\Omega \times A^\Omega \times A_{\xi,T}^\Omega \rightarrow [0;1]$ where $\tilde{s}_{t+1} \sim T_\xi^\Omega(\cdot|s_t, a_t, a_t^{\xi,T})$ is applied as substitute of the main transition function of the environment T^Ω .

The next state s_{t+1} (after transition)

Via an additional transition functions $T_{\xi+}^\Omega : S^\Omega \times S^\Omega \times A_{\xi,S+}^\Omega \rightarrow [0;1]$ where $s_{t+1} \sim T_{\xi+}^\Omega(\cdot|s_{t+1}, a_t^{\xi,S+})$ is applied after the main transition function of the environment and before the next decision a_{t+1} of the agent is taken.

The perturbations on the two first types of elements (observation and action) require just to modify a vector which will be fed as input of another function, so they are easy to implement in any environment. The perturbations on the three last types of elements (state, transition function and next state) are more complex and require to modify the environment itself, either by being able to modify the state with an additional transition function, or being able to modify the main transition function itself by incorporating the effect of the adversary action.

Here and in the following, we use the term **perturb** to denote direct modification of an element by an adversarial action a_t^ξ . For example, direct perturbation of an observation x_t , perturbation of a state s_t , perturbations of an action a_t . We do not use the term perturbations for indirect modifications, for example by directly perturbing an observation x_t , the action a_t chosen by the agent could be modified, this new action cannot be seen as a perturbed action but results from the application of the policy π of the agent given a perturbed observation x'_t .

In the following we use the term **disturb** to denote any perturbation of one of the following elements : action, state, transition function or next state. More generally the term disturb is used to denote perturbations that modifies the dynamics of the environment.

4.2 Altered POMDP Component

Following the two main types of alterations ϕ that are discussed in previous section, the main axis of the taxonomy of approaches concerns the impact on the POMDP of actions that are emitted by adversary agents during training of π . Given adversarial elements defined in the previous section, we specify each possible perturbation independently to discuss each specific adversarial impact on the POMDP.

Here and in the following, we use the term **alter** to denote the modification of a component of the POMDP from the POV of the agent. For example, adding an adversarial attack that perturb the observations is an alteration of the observation function O^Ω of the POMDP from the POV of the agent. Alternatively, adding an adversarial attack that perturb the action, the state, the transition function or the next state, is an alteration of the transition function T^Ω of the POMDP (dynamics of the environment) from the POV of the agent.

4.2.1 Alteration of the Observations Function O^Ω

The first type of component alteration is the alteration of the observation function O^Ω of the POMDP. Directly inspired from adversarial attacks in supervised machine learning, many methods are designed to modify the inputs that are perceived by the protagonist agent π . The principle is to modify the input vector of an agent, which can correspond for instance to the outputs of a sensor of a physical agent, like an autonomous vehicle. The observation is perturbed before the agent take any decision, so the agent get the perturbed observation and can be fooled.

More formally, in the setting of an observation attack, the adversary ξ acts to produce a perturbed observation x'_t before it is fed as input to π , by perturbing the observation x_t via the specific perturbation function $O^\xi(x'_t|x_t, a_t^{\xi, X})$ introduced in Section 4.1.

In that case, ξ can be regarded as an adversary agent that acts by emitting adversarial actions $a_t^{\xi, X} \sim \xi(\cdot|s_t, x_t)$ with $a_t^{\xi, X} \in A_{\xi, X}^\Omega$, given π in a POMDP defined as $\Omega^\pi = (S^\Omega, A_{\xi, X}^\Omega, T^{\pi, \Omega}, R_\xi^\Omega, X^\Omega, O^\Omega)$, where sampling $s_{t+1} \sim T^{\pi, \Omega}(\cdot|s_t, a_t^{\xi, X})$ is performed in four steps, as shown in Algorithm 1.

Algorithm 1 $s_{t+1} \sim T^{\pi, \Omega}(\cdot | s_t, a_t^{\xi, X})$

Input $s_t, a_t^{\xi, X}$	▷ state and adversary action
1: sample $x_t \sim O^{\Omega}(\cdot s_t)$	▷ observation
2: sample $x'_t \sim O^{\xi}(\cdot x_t, a_t^{\xi, X})$	▷ perturbed observation
3: sample $a_t \sim \pi(\cdot x'_t)$	▷ agent action
4: sample $s_{t+1} \sim T^{\Omega}(\cdot s_t, a_t)$	▷ next state after transition
return s_{t+1}	

Reversely, agent π acts on an altered POMDP $\Omega^{\xi} = (S^{\Omega}, A^{\Omega}, T^{\Omega}, R^{\Omega}, X^{\Omega}, O^{\xi, \Omega})$ where the input observation $x'_t \sim O^{\xi, \Omega}(\cdot | s_t)$ is performed in three steps, as shown in Algorithm 2.

Algorithm 2 $x'_t \sim O^{\xi, \Omega}(\cdot | s_t)$

Input s_t	▷ state
1: sample $x_t \sim O^{\Omega}(\cdot s_t)$	▷ observation
2: sample $a_t^{\xi, X} \sim \xi(\cdot s_t, x_t)$	▷ adversary action
3: sample $x'_t \sim O^{\xi}(\cdot x_t, a_t^{\xi, X})$	▷ perturbed observation
return x'_t	

Figure 3 presents a flowchart illustrating how the observation perturbation integrates into the POMDP.

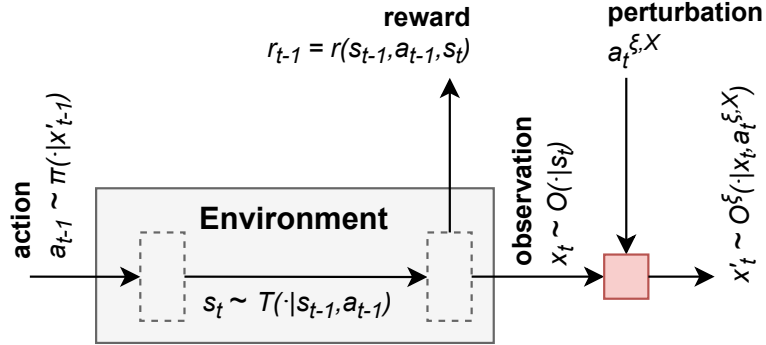


Fig. 3: Flowchart of the perturbation of the observation

Following this, the probability of an adversary-augmented transition $\tilde{\tau}_t = (s_t, x_t, a_t, a_t^{\xi}, x'_t, a'_t, \tilde{s}_t, \tilde{x}_t, \tilde{s}_{t+1}, \tilde{x}_{t+1}, s_{t+1})$ given past $\tilde{\tau}_{0:t-1}$, is given by:

$$\begin{aligned} \pi^{\xi, \Omega}(\tau_t | s_t) = & O^\Omega(x_t | s_t) \xi(a_t^{\xi, X} | s_t, x_t) O^\xi(x'_t | x_t, a_t^{\xi, X}) \pi(a_t | x'_t) \delta_{a_t}(a'_t) \\ & \delta_{s_t}(\tilde{s}_t) O^\Omega(\tilde{x}_t | \tilde{s}_t) T^\Omega(\tilde{s}_{t+1} | \tilde{s}_t, a'_t) O^\Omega(\tilde{x}_{t+1} | s_{t+1}) \delta_{\tilde{s}_{t+1}}(s_{t+1}) \end{aligned}$$

where δ_x stands for a Dirac distribution centered on x .

4.2.2 Alteration of the Transition Function T^Ω (Environment Dynamics)

The other type of component alteration is the alteration of the transition function T^Ω of the POMDP (altering the dynamics of the environment). The principle is to modify effects of actions of the protagonist in the environment. For example, this can include moving or modifying the behavior of some physical objects in the environment, like modifying the positions or speed of some vehicles in autonomous driving simulator, or modifying the way the protagonist's actions are affecting the environment (e.g. by amplifying or reversing actions).

This is done by emitting adversarial actions A_ξ^Ω , that are allowed by the environment Ω through a specific adversary function A^ξ , $T_{\xi^-}^\Omega$, T_ξ^Ω or $T_{\xi^+}^\Omega$, creating an altered transition function $T^{\xi, \Omega}$ for the protagonist actions. In that setting, four types of adversaries can be considered:

Transition Perturbation

In this setting, the process begins with the agent in an initial state. The agent then chooses an action, which is applied to the environment. This should lead to a transition to a new state, according to the environment's transition function. However, this transition function is perturbed, effectively altering the dynamics of the environment. Resulting in a different new subsequent state than if the transition had not been perturbed.

For instance, in the context of an autonomous vehicle, the vehicle might decide to change lanes (action) based on the existing traffic setup (state). As this action leads to a transition, the behavior of surrounding vehicles is unpredictably modified (perturbed transition). Consequently, the vehicle emerges in a new traffic configuration (next state) that is different from what would typically result from the chosen action.

This process introduces variability into the environment's dynamics by directly changing the environment's inherent transition function.

More formally, the adversary ξ acts to induce an altered next state s'_{t+1} by modifying the transition function itself, replacing it with the perturbed transition function $T_\xi^\Omega(s_{t+1} | (s_t, a_t), a_t^{\xi, T})$ introduced in Section 4.1. In that case, ξ can be regarded as an agent that acts by emitting adversarial actions $a_t^{\xi, T} \sim \xi(\cdot | s_t, x_t, a_t)$, given π in a POMDP defined as: $\Omega^\pi = ((S^\Omega, A^\Omega), A_{\xi, T}^\Omega, T^{\pi, \Omega}, R_\xi^\Omega, X^\Omega, O^\Omega)$, where sampling $(s_{t+1}, a_{t+1}) \sim T^{\pi, \Omega}(\cdot | (s_t, a_t), a_t^{\xi, T})$ is performed in three steps, as shown in Algorithm 3.

Algorithm 3 $(s_{t+1}, a_{t+1}) \sim T^{\pi, \Omega}(\cdot | (s_t, a_t), a_t^{\xi, T})$

Input $s_t, a_t, a_t^{\xi, T}$ ▷ state, agent action and adversary action
1: sample $s_{t+1} \sim T_{\xi}^{\Omega}(\cdot | s_t, a_t, a_t^{\xi, T})$ ▷ next state after perturbed transition
2: sample $x_{t+1} \sim O^{\Omega}(\cdot | s_{t+1})$ ▷ next observation
3: sample $a_{t+1} \sim \pi(\cdot | x_{t+1})$ ▷ next agent action
return s_{t+1}, a_{t+1}

Figure 4 presents a flowchart illustrating how the transition perturbation integrates into the POMDP.

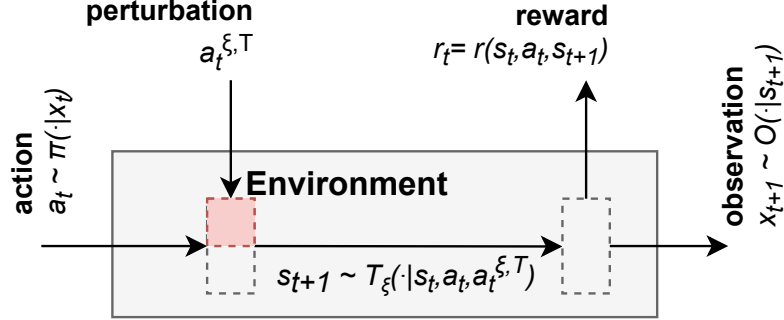


Fig. 4: Flowchart of the perturbation of the transition function

Current State Perturbation

In this setting, the process begins with the agent in an initial state. The agent then chooses an action to be applied within the environment. However, before this action is applied, the current state is subjected to a perturbation. This perturbation alters the initial state, leading to a modified state in which the chosen action is applied. The application of the action in this perturbed state results in a transition, resulting in a new subsequent state according to the environment’s transition function.

For example, consider an autonomous vehicle deciding to change lanes (action) based on the prevailing traffic configuration (state). Prior to executing this maneuver, the traffic configuration is altered (perturbed state), such as by adjusting the positions of nearby vehicles. Consequently, when the vehicle executes its lane change, it does so in this adjusted traffic scenario, leading to a different traffic configuration (next state) than if the original state had not been modified.

This process introduces variability into the environment’s dynamics without necessitating a direct modification of the environment’s transition function.

More formally, the adversary ξ acts to induce an altered next state s'_{t+1} by perturbing the state before the transition function via the prior transition function $T_{\xi^-}^{\Omega}(\tilde{s}_t|s_t, a_t^{\xi, S})$ introduced in Section 4.1. In that case, ξ can be regarded as an agent that acts by emitting adversarial actions $a_t^{\xi, S} \sim \xi(\cdot|s_t, x_t, a_t)$, given π in a POMDP defined as: $\Omega^{\pi} = ((S^{\Omega}, A^{\Omega}), A_{\xi, S}^{\Omega}, T^{\pi, \Omega}, R_{\xi}^{\Omega}, X^{\Omega}, O^{\Omega})$, where sampling $(s_{t+1}, a_{t+1}) \sim T^{\pi, \Omega}(\cdot|(s_t, a_t), a_t^{\xi, S})$ is performed in four steps, as shown in Algorithm 4.

Algorithm 4 $(s_{t+1}, a_{t+1}) \sim T^{\pi, \Omega}(\cdot|(s_t, a_t), a_t^{\xi, S})$

Input $s_t, a_t, a_t^{\xi, S}$ ▷ state, agent action and adversary action
1: sample $\tilde{s}_t \sim T_{\xi^-}^{\Omega}(\cdot|s_t, a_t^{\xi, S})$ ▷ perturbed state
2: sample $s_{t+1} \sim T^{\Omega}(\cdot|\tilde{s}_t, a_t)$ ▷ next state after transition
3: sample $x_{t+1} \sim O^{\Omega}(\cdot|s_{t+1})$ ▷ next observation
4: sample $a_{t+1} \sim \pi(\cdot|x_{t+1})$ ▷ next agent action
return s_{t+1}, a_{t+1}

Figure 5 presents a flowchart illustrating how the current state perturbation integrates into the POMDP.

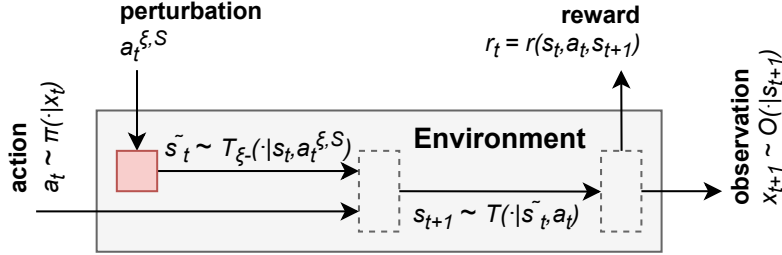


Fig. 5: Flowchart of the perturbation of the current state

Next State Perturbation

In this setting, the process begins with the agent in an initial state. The agent chooses an action which is then applied in the environment. This leads to transition to a new subsequent state, according to the environment's transition function. However, before the agent can choose its next action, this new state is perturbed.

For instance, in the case of an autonomous vehicle, the vehicle might choose to change lanes (action) based on the current traffic configuration (state). After the action is executed, the vehicle finds itself in a new traffic configuration (next state). Before

choosing the next action, this new state is perturbed, for example, by altering the positions of surrounding vehicles. This means the vehicle now faces a modified traffic configuration (perturbed next state) from which it must decide its next move.

This process introduces variability into the environment’s dynamics without necessitating a direct modification of the environment’s transition function.

The key difference between perturbing the current state and perturbing the next state lies in the agent’s awareness of its situation. In current state perturbation, the agent lacks true knowledge of its precise state when choosing the action because this state is modified just before the action is applied. However, in next state perturbation, the agent has full awareness of its current state when choosing the action.

More formally, the adversary ξ acts to produce an altered next state s'_{t+1} by perturbing the state after the transition function via the posterior transition function $T_{\xi+}^{\Omega}(\tilde{s}_{t+1}|s_{t+1}, a_t^{\xi, S+})$ introduced in Section 4.1. In that case, ξ can be regarded as an agent that acts by emitting adversarial actions $a_t^{\xi, S+} \sim \xi(\cdot|s_t, x_t)$, given π in a POMDP defined as: $\Omega^{\pi} = (S^{\Omega}, A_{\xi, S+}^{\Omega}, T^{\pi, \Omega}, R_{\xi}^{\Omega}, X^{\Omega}, O^{\Omega})$, where sampling $s_{t+1} \sim T^{\pi, \Omega}(\cdot|s_t, a_t^{\xi, S+})$ is performed in three steps, as shown in Algorithm 5.

Algorithm 5 $s_{t+1} \sim T^{\pi, \Omega}(\cdot|s_t, a_t^{\xi, S+})$

Input $s_t, a_t^{\xi, S+}$	▷ state, adversary action
1: sample $\tilde{s}_t \sim T_{\xi+}^{\Omega}(\cdot s_t, a_t^{\xi, S+})$	▷ perturbed state
2: sample $\tilde{x}_t \sim O^{\Omega}(\cdot \tilde{s}_t)$	▷ observation
3: sample $a_t \sim \pi(\cdot x_t)$	▷ agent action
4: sample $s_{t+1} \sim T^{\Omega}(\cdot \tilde{s}_t, a_t)$	▷ next state after transition
return s_{t+1}	

Figure 6 presents a flowchart illustrating how the next state perturbation integrates into the POMDP.

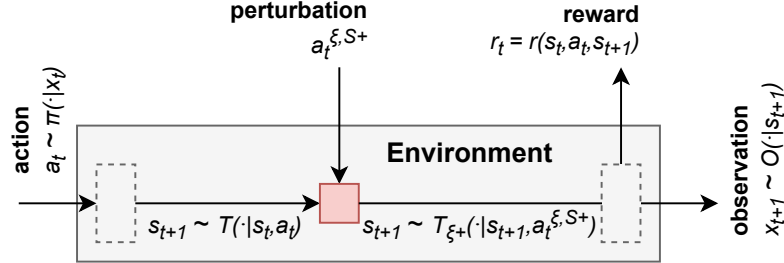


Fig. 6: Flowchart of the perturbation of the next state

Action Perturbation

In this setting, the process starts with the agent in an initial state. The agent chooses an action, which is intended to be applied in the environment. However, before this action can be applied, it undergoes a perturbation, resulting in a perturbed action. This perturbed action is then applied, leading to transition to a new state according to the environment’s transition function.

For instance, consider an autonomous vehicle that decides to steer at an angle of α (action) based on the current traffic configuration (state). Before the steering action is executed, it is perturbed, so the actual steering angle applied to the vehicle becomes $\alpha + \epsilon$ (perturbed action). As a result, the vehicle transitions into a new traffic configuration (next state) that reflects the outcome of the perturbed steering action.

This process introduces variability into the environment’s dynamics without necessitating a direct modification of the environment’s transition function or modification of the state of the environment. However, this approach to modifying dynamics, while introducing variability, is confined to the scope of action perturbation, limiting the diversity of potential dynamics alterations.

More formally, the adversary ξ acts to induce an altered next state s'_{t+1} by perturbing the action decided by the agent $a_t \sim \pi(\cdot|x_t)$ via the specific perturbation function $A^\xi(a'_t|a_t, a_t^{\xi,A})$ introduced in Section 4.1.

In that case ξ can be regarded as an agent that acts by emitting adversarial actions $a_t^{\xi,A} \sim \xi(\cdot|s_t, x_t, a_t)$, given π in a POMDP defined as: $\Omega^\pi = ((S^\Omega, A^\Omega), A_{\xi,A}^\Omega, T^{\pi,\Omega}, R_\xi^\Omega, X^\Omega, O^\Omega)$, where sampling $(s_{t+1}, a_{t+1}) \sim T^{\pi,\Omega}(\cdot|(s_t, a_t), a_t^{\xi,A})$ is performed in four steps, as shown in Algorithm 6.

Algorithm 6 $(s_{t+1}, a_{t+1}) \sim T^{\pi,\Omega}(\cdot|(s_t, a_t), a_t^{\xi,A})$

Input $s_t, a_t, a_t^{\xi,A}$	▷ state, agent action and adversary action
1: sample $a'_t \sim A^\xi(\cdot a_t, a_t^{\xi,A})$	▷ perturbed agent action
2: sample $s_{t+1} \sim T^\Omega(\cdot s_t, a'_t)$	▷ next state after transition
3: sample $x_{t+1} \sim O^\Omega(\cdot s_{t+1})$	▷ next observation
4: sample $a_{t+1} \sim \pi(\cdot x_{t+1})$	▷ next agent action
return s_{t+1}, a_{t+1}	

Figure 7 presents a flowchart illustrating how the action perturbation integrates into the POMDP.

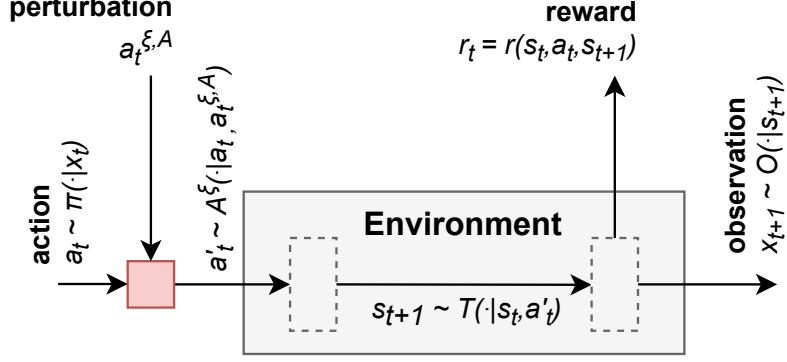


Fig. 7: Flowchart of the perturbation of the action

Reversely, we can gather the point of view of the protagonist agent π in a single example combining all four possible attacks just described by denoting the adversaries ξ_A , ξ_{S-} , ξ_T and ξ_{S+} . The agent π acts on an altered POMDP $\Omega^\xi = (S^\Omega, A^\Omega, T^{\xi, \Omega}, R^\Omega, X^\Omega, O^\Omega)$, where $s_{t+1} \sim T^{\xi, \Omega}(\cdot | s_t, a_t)$ is performed in eleven steps, as shown in Algorithm 7.

Algorithm 7 $s_{t+1} \sim T^{\xi, \Omega}(\cdot | s_t, a_t)$

Input s_t, a_t	▷ state and agent action
1: sample $x_t \sim O^\Omega(\cdot s_t)$	▷ observation
2: sample $a_t^{\xi, A} \sim \xi_A(\cdot s_t, x_t, a_t)$	▷ adversary action A
3: sample $a_t^{\xi, S} \sim \xi_{T-}(\cdot s_t, x_t, a_t)$	▷ adversary action S
4: sample $a'_t \sim A^\xi(\cdot a_t, a_t^{\xi, A})$	▷ perturbed action
5: sample $\tilde{s}_t \sim T_{\xi-}^\Omega(\cdot s_t, a_t^{\xi, S})$	▷ perturbed state
6: sample $\tilde{x}_t \sim O^\Omega(\cdot \tilde{s}_t)$	▷ observation of the perturbed state
7: sample $a_t^{\xi, T} \sim \xi_T(\cdot \tilde{s}_t, \tilde{x}_t, a'_t)$	▷ adversary action T
8: sample $\tilde{s}_{t+1} \sim T_{\xi}^\Omega(\cdot \tilde{s}_t, a'_t, a_t^{\xi, T})$	▷ next state after perturbed transition
9: sample $\tilde{x}_{t+1} \sim O^\Omega(\cdot \tilde{s}_{t+1})$	▷ observation of the next state
10: sample $a_t^{\xi, S+} \sim \xi_{T+}(\cdot \tilde{s}_{t+1}, \tilde{x}_{t+1})$	▷ adversary action S+
11: sample $s_{t+1} \sim T_{\xi+}^\Omega(\cdot \tilde{s}_{t+1}, a_t^{\xi, S+})$	▷ perturbed next state
return s_{t+1}	

Following this, the probability of an adversary-augmented transition $\tilde{\tau}_t = (s_t, x_t, a_t, a_t^{\xi, A}, a_t^{\xi, S}, a'_t, \tilde{s}_t, \tilde{x}_t, \tilde{s}_{t+1}, \tilde{x}_{t+1}, s_{t+1})$ given past $\tilde{\tau}_{0:t-1}$, is given by:

$$\begin{aligned}
\pi^{\xi, \Omega}(\tilde{\tau}_t | s_t) &= O^\Omega(x_t | s_t) \delta_{x_t}(x'_t) \pi(a_t | x'_t) \xi_A(a_t^{\xi, A} | s_t, x_t, a_t) A^\xi(a'_t | a_t, a_t^{\xi, A}) \\
&\xi_{S^-}(a_t^{\xi, S} | s_t, x_t, a_t) T_{\xi^-}^\Omega(\tilde{s}_t | s_t, a_t^{\xi, S}) O^\Omega(\tilde{x}_t | \tilde{s}_t) \xi_T(a_t^{\xi, T} | \tilde{s}_t, \tilde{x}_t, a'_t) T_{\xi}^\Omega(\tilde{s}_{t+1} | \tilde{s}_t, a'_t, a_t^{\xi, T}) \\
&O^\Omega(\tilde{x}_{t+1} | \tilde{s}_{t+1}) \xi_{S^+}(a_t^{\xi, S^+} | \tilde{s}_{t+1}, \tilde{x}_{t+1}) T_{\xi^+}^\Omega(s_{t+1} | \tilde{s}_{t+1}, a_t^{\xi, S^+})
\end{aligned}$$

In all settings, the reward functions R^Ω and R_ξ^Ω are defined on environment transitions (s_t, a_t, s_{t+1}) .

4.3 Adversarial Objective

Adversarial attacks in RL are strategically designed to compromise specific aspects of agent behavior or environment dynamics. In general, they aim to prevent the agent from acting optimally, but the attacks vary in their objectives and methodologies. Even if the general goal of any adversarial attack is to reduce the performances of the agent, methods to achieve this can primarily have different objective function, for specific performance reductions.

4.3.1 Deviate Policy

The primary goal is to deviate the agent from its initial, typically optimal, policy. We can deviate the policy to make it diverge from the original policy : in that case the adversary ξ is designed to maximize the agent's expected loss over the pairs of policy given perturbed observation x' and original observation x , this is often referred as **untargeted attacks**. Or, we can also deviate the policy to make it converge to a target policy: in that case the adversary ξ is designed to minimize the agent's expected loss over the pairs of policy given perturbed observation x' and another policy g given original observation x , this is often referred as **targeted attacks**.

For adversarial attacks that alter the observation function, assuming the following shorthand notations: $\mathbb{E}_x = \mathbb{E}_{x \sim X}$; $\mathbb{E}_a = \mathbb{E}_{a^\xi \sim \xi(\cdot | x)}$; $\mathbb{E}_{x'} = \mathbb{E}_{x' \sim O^\xi(\cdot | x, a^\xi), \|x' - x\| < \varepsilon}$, the objective function for untargeted attacks (policy divergence) is :

$$\xi^* = \arg \max_{\xi} \mathbb{E}_x \mathbb{E}_a \mathbb{E}_{x'} \left[\mathcal{L}(\pi(x'), \pi(x)) \right]$$

The objective function for targeted attacks (policy convergence) is :

$$\xi^* = \arg \min_{\xi} \mathbb{E}_x \mathbb{E}_a \mathbb{E}_{x'} \left[\mathcal{L}(\pi(x'), g(x)) \right]$$

These formulations seek for the optimal adversarial strategy ξ^* that maximizes (resp. minimizes) the expected loss \mathcal{L} , computed over a distribution of original observations x , actions a^ξ according to the adversary policy, and perturbed observations x' resulting from the altered observation function O^ξ , constrained by the condition that the perturbation in observation is less than ε . The loss \mathcal{L} measures the difference between the policy's output over the perturbed observations $\pi(x')$ and the policy's output over the original observations $\pi(x)$ (resp. the target policy's output over the original observations $g(x)$).

For adversarial attacks that alter the dynamics of the environment, assuming the following shorthand notations: $\mathbb{E}_{s,a} = \mathbb{E}_{(s,a) \sim (S,A)}$; $\mathbb{E}_x = \mathbb{E}_{x \sim O^\Omega(\cdot|s)}$; $\mathbb{E}_{a^\xi} = \mathbb{E}_{a^\xi \sim \xi(\cdot|x)}$; $\mathbb{E}_{s_{t+1}} = \mathbb{E}_{s_{t+1} \sim T^\Omega(\cdot|s,a)}$; $\mathbb{E}_{x_{t+1}} = \mathbb{E}_{x_{t+1} \sim O(\cdot|s_{t+1})}$; $\mathbb{E}_{\tilde{s}_{t+1}} = \mathbb{E}_{\tilde{s}_{t+1} \sim T^{\xi,\Omega}(\cdot|s,a), \|\tilde{s}_{t+1} - s_{t+1}\| < \varepsilon}$; $\mathbb{E}_{\tilde{x}_{t+1}} = \mathbb{E}_{\tilde{x}_{t+1} \sim O(\cdot|\tilde{s}_{t+1})}$, the objective function for untargeted attacks (policy divergence) is :

$$\xi^* = \arg \max_{\xi} \mathbb{E}_{s,a} \mathbb{E}_x \mathbb{E}_{a^\xi} \mathbb{E}_{s_{t+1}} \mathbb{E}_{x_{t+1}} \mathbb{E}_{\tilde{s}_{t+1}} \mathbb{E}_{\tilde{x}_{t+1}} \left[\mathcal{L}(\pi(\tilde{x}_{t+1}), \pi(x_{t+1})) \right]$$

The objective function for targeted attacks (policy convergence) is :

$$\xi^* = \arg \min_{\xi} \mathbb{E}_{s,a} \mathbb{E}_x \mathbb{E}_{a^\xi} \mathbb{E}_{s_{t+1}} \mathbb{E}_{x_{t+1}} \mathbb{E}_{\tilde{s}_{t+1}} \mathbb{E}_{\tilde{x}_{t+1}} \left[\mathcal{L}(\pi(\tilde{x}_{t+1}), g(x_{t+1})) \right]$$

Here, ξ^* is the optimal adversarial strategy that maximizes (resp. minimizes) the expected loss, considering the altered state dynamics and resulting observations, with the constraint that the perturbation in the state is less than ε . The expectations are over the distribution of original states s and their observations x , adversarial actions a^ξ , perturbed states \tilde{s}_{t+1} from the modified transition function $T^{\xi,\Omega}$, and observations \tilde{x}_{t+1} from the perturbed states. The loss \mathcal{L} measures the difference between the policy's output over the observations of perturbed next states $\pi(\tilde{x}_{t+1})$ and the policy's output over the observations of original non perturbed next states $\pi(x_{t+1})$ (resp. the target policy's output over the observations of original non perturbed next states $g(x_{t+1})$).

Even if the goal of applying these attacks can be to reduce the performances of the agent, the attacks themselves are designed to maximize the divergence (resp. convergence) of the policy, effectively causing the policy to produce significantly different actions or decisions based on the manipulated environment dynamics and observations.

4.3.2 Reward Minimization

In contrast, some adversarial attacks focus on leading the agent to less favorable states or decisions, thereby minimizing the total expected reward the agent accrues. These attacks are often targeted and seek for reduction of the efficacy or efficiency of the agent's behavior by altering its reward acquisition.

The objective function optimized in such attacks, subject to the relevant perturbation constraints, is:

$$\xi^* = \arg \min_{\xi} \mathbb{E}_{\tilde{\tau} \sim \pi^{\xi,\Omega}(\tilde{\tau})} \left[R(\tilde{\tau}) \right] \tilde{\tau} \text{ subject to } \|x' - x\| < \varepsilon \text{ or } \|\tilde{s}_{t+1} - s_{t+1}\| < \varepsilon$$

Here, ξ^* represents the adversarial strategy aimed at minimizing the agent's total expected reward. The expectation is taken over the trajectories τ sampled according to a policy π perturbed by the adversary in the environment Ω . The function $R(\tau)$ calculates the discounted sum of rewards for each trajectory, with the goal of the adversary being to minimize this quantity through interventions, while adhering to the specified constraints on the perturbations. For attacks altering the observation, the difference between the perturbed observation x' and the original observation x is constrained such that $\|x' - x\| < \varepsilon$. Similarly, for attacks that alter the dynamics of the environment, the perturbation in the state is constrained with $\|\tilde{s}_{t+1} - s_{t+1}\| < \varepsilon$.

In practice, it is hard to verify $\|\tilde{s}_{t+1} - s_{t+1}\| < \varepsilon$ since once the transition function applied on the next state get, in most of the simulation environment available, it is not possible to redo the transition with some other inputs to get the alternative next state, to be able to compare them. Therefore often, the actually verified constraint is $\|a_t^\xi\| < \varepsilon$, this makes it hard to compare different Dynamic Attacks that do not alter the same element s_t , a_t , T^Ω , or s_{t+1} .

4.3.3 Others

Some methods have other objectives, for example to lead the agent to a specific target state. The objective is then to minimize the distance between the current state and the target state. Some other specific objectives can exist.

4.4 Knowledge Requirement

In the realm of adversarial attacks against DRL agents, the extent and nature of the attacker’s knowledge about the agent significantly influence the strategy and effectiveness of the attack. Broadly, these can be categorized into White Box and Black Box approaches, each with its own set of strategies, challenges, and considerations.

4.4.1 White Box

In this scenario, the adversary has complete knowledge of the agent’s architecture, parameters, and training data. This scenario represents the most informed type of attacks, where the adversary has access to all the inner workings of the agent, including its policy, value function, and possibly even the environment model.

- Policy and Model Access: The adversary knows the exact policy and decision-making process of the agent. This includes access to the policy’s parameters, algorithm type, and architecture. In model-based RL, the attacker might also know the transition dynamics and reward function.

- Optimization and Perturbation: With complete knowledge, the attacker can craft precise and potent perturbations to the agent’s inputs or environment to maximize the deviation from desired behaviors or minimize rewards. They can calculate the exact gradients or other relevant information needed to optimize their attack strategy.

- Challenges and Implications: While white box attacks represent an idealized scenario with maximal knowledge, they provide a comprehensive framework for testing the agent’s robustness. By simulating the most extreme conditions an agent could face, developers can identify and reinforce potential vulnerabilities, leading to policies that are not only effective but also resilient to a wide range of scenarios, including unexpected environmental changes. This approach is particularly valuable in safety-critical applications where ensuring reliability against all possible disturbances is crucial.

4.4.2 Black Box

In this scenario, the adversary has limited or no knowledge of the internal workings of the agent. They may not know the specific policy, parameters, or architecture of the RL agent. Instead, they must rely on observable behaviors or outputs to infer

information and craft their attacks.

- **Observational Inference:** The attacker observes the agent’s actions and possibly some aspects of the state transitions to infer patterns, weaknesses, or predict future actions. This process often involves probing the agent with different inputs and analyzing the outputs.

- **Surrogate Models and Transferability:** Attackers might train a surrogate model to approximate the agent’s behavior or policy. If an attack is successful on the surrogate, it might also be effective on the target agent due to transferability, especially if both are trained in similar environments or tasks.

- **Challenges and Implications:** The use of black box methods in enhancing robustness is not directly about realism of adversarial intent but rather about preparing for a variety of uncertain conditions and environmental changes. These methods encourage the development of general defense mechanisms that improve the agent’s adaptability and resilience. While the adversarial mindset might not reflect the typical operational challenges, the diversity and unpredictability of black box approaches help ensure that RL systems are robust not only against potential adversaries but also against a wide array of non-adversarial issues that could arise in dynamic and uncertain environments.

Both white and black box attacks paradigms play crucial roles in the study and development of adversarial strategies in RL. They help researchers and practitioners understand the spectrum of threats and devise more robust algorithms and defenses. By considering these different knowledge scenarios, one can better prepare RL agents to withstand or recover from adversarial attacks in various real-world applications.

4.5 Category of Approach

This section delineates the various methodologies utilized in crafting adversarial attacks, each with distinct strategies and theoretical underpinnings. It primarily divides into direct optimization-based and adversarial policy learning approaches.

4.5.1 Direct Optimization Based Approaches

They focus on directly manipulating the input or parameters of a model to induce misbehavior. These methods are subdivided into first-order and zeroth-order techniques, depending on the availability and usage of gradient information.

First Order Optimization approaches (White Box) : Gradient Attacks

They utilize the gradient information of the model to craft adversarial examples, efficiently targeting the model’s weaknesses. Common in white-box scenarios, gradient attacks are powerful when model internals are accessible.

Zeroth Order Optimization approaches (Black Box)

Or derivative-free methods, optimize the adversarial objective without requiring gradient information, making them suitable for black-box scenarios. Techniques include simulated annealing, genetic algorithms, and random search.

4.5.2 Adversarial Policy Learning Based Approaches

These approaches involve training a separate model or policy to generate adversarial attacks. The adversarial model learns an optimal attack strategy through interaction with the target system, often using RL techniques. To train an Adversarial Policy (AP), optimization methods are used and could also be classified as First and Zeroth Order methods, but on the contrary to direct optimization methods, the optimization is used to train the adversary, not to directly craft the perturbation. Adversarial Policy Learning Based Approaches can be divided into two categories :

Classical Adversarial Policies

Learned via RL or any other method, only need a black box access to the model of the agent since they are policies sufficient in themselves.

Augmented Adversarial Policies

Learned via RL or any other approach, are augmented either with a white Box access to the agent’s model during training or inference phase, or with some Direct Optimization method are added besides the Adversarial Policy to improve its performances.

In the following sections, we will use this taxonomy as a framework to examine recent research on adversarial examples for DRL. Section 5.1 focuses on input-space perturbations, and Section 5.2 on environment-space perturbations.

5 Adversarial Attacks

In this section, we conduct a comprehensive review of contemporary adversarial attacks as documented in current literature, presented in a hierarchical, tree-like structure (refer to Figure 1). The review categorizes these attacks first based on the type of alteration induced in the POMDP: either Observation Alteration or Dynamic Alteration. Next, the categorization considers the underlying objective driving these attacks, which could be either to Deviate Policy or Minimize Reward. Lastly, the classification focuses on the computational approach employed: Direct Optimization (First or Zeroth Order) or Adversarial Policy Learning. For each method in this classification tree, we will provide a detailed description, ensuring to consistently include the following critical information: the nature of the perturbation support (whether it’s an observation, state, action, or transition function), the level of knowledge about the model required to execute the attack (white-box or black-box), and any specific constraints or potential limitations associated with the method.

5.1 Observation Alteration Attacks

This section delves into the analysis of Observation Alteration Attacks targeting RL agents. These attacks specifically modify the observation function in the POMDP framework. Such methods are instrumental in simulating sensor errors in an agent, creating discrepancies between the agent’s perceived observations and the actual underlying state. These techniques can be particularly beneficial during an agent’s training phase, enhancing its resilience to potential observation discrepancies that

might be encountered in real-world deployment scenarios. Observation Alteration Attacks generate a perturbation $a_\epsilon^{\xi, X}$ for a given observation x , resulting in a perturbed observation $x' = x + a_\epsilon^{\xi, X}$. The perturbation $a_\epsilon^{\xi, X}$ is constrained within an ϵ -ball of a specified norm $p \in L_1, L_2, \dots, L_\infty$. This constraint can be achieved from any arbitrary perturbation $a^{\xi, X}$ by computing $a_\epsilon^{\xi, X} = \frac{\epsilon}{\|a^{\xi, X}\|_p} \cdot a^{\xi, X}$. Alternatively, in the context of a L_0 ϵ -ball, $a_\epsilon^{\xi, X}$ is defined by assigning the top- ϵ values of $a^{\xi, X}$ as 1 and setting all other values to 0.

5.1.1 Deviate Policy

In the field of adversarial attacks on observations, most methods developed primarily focus on optimizing the deviation of the policy. These methods function by crafting a perturbed observation x' that replaces the original observation x . Consequently, the policy π generates a different output $\pi(x') \neq \pi(x)$. In the untargeted scenario, the goal is to maximize divergence from the original policy; this is achieved by maximizing a specific loss function between the policy output on the altered observation and the original observation, formulated as $\arg \max_{x'} L(\pi(x'), \pi(x))$. Conversely, in targeted attacks, the objective is to guide the policy towards a particular behavior. This is done by minimizing a defined loss between the policy output on the altered observation and a target policy g on the original observation, expressed as $\arg \min_{x'} L(\pi(x'), g(x))$. While the primary focus of these optimization functions is the deviation of the policy, this often results in a corresponding reduction in the reward garnered by the agent. Although Adversarial Policy Learning Based Methods could theoretically be employed to create observation attacks intended to deviate policy, the prevailing methods in practice are predominantly Direct Optimization Methods.

Direct Optimization Methods refer to techniques that directly compute an adversarial perturbation directly with optimization approaches. These approaches include gradient descent, evolutionary methods, stochastic optimization, among others. They are particularly effective for generating perturbations in observations with the aim of altering the policy's behavior. A key advantage of these methods is their ability to be applied directly to a given agent model, without the necessity for extensive prior knowledge or preliminary computations. However, it is important to note that some of these methods might entail significant computational resources for each perturbation calculation.

First Order Optimization Methods: Gradient Attacks (White Box)

They are methods initially introduced in the context of supervised classification. They utilize the gradient of the attacked model to compute a perturbation $a^{\xi, X}$ for a given input x , thereby crafting a perturbed input x' . Consequently, these methods require white-box access to the model being attacked. In the realm of supervised classification, they are typically defined using the general formula:

$$x' = x + a^{\xi, X} \quad \text{with} \quad a^{\xi, X} = \epsilon \times \dots \nabla_x L(f(x), y) \dots$$

Here, ε represents the magnitude of the perturbation, $f(x)$ is the model output, y denotes the ground truth label for untargeted attacks or the target class for targeted attacks, and L is a loss function (often the same one used in training, but not exclusively). The term $\nabla_x L$ signifies the gradient of the loss function L with respect to the input x , and the “...” indicates that additional operations can be applied to this core equation to tailor the update function to the specific optimization problem at hand. When adapted to RL, the formula essentially remains unchanged, except $f(x)$ is replaced by $\pi(x)$, the output of the agent’s policy function. In this context, y no longer represents the ground truth but rather the current action a for untargeted attacks, or a targeted action for targeted attacks. Numerous gradient attack methods exist, with the most notable being FGSM and its extensions (BIM, PGD, C&W, DeepFool, ...), as well as JSMA and its extensions (XSMA, VFGA, ...). All these methods, initially designed for supervised classification, are applicable in RL. Their primary objective is to deviate the agent’s policy by creating adversarial observations. As they are designed to generate minimal perturbations around a given observation, they are generally suited for agents and environments with continuous observation spaces, such as images, feature vectors, and signals. These attacks employ first-order optimization methods as they utilize the direction directly provided by the model’s gradient, thus categorizing them as white-box methods, which necessitate complete knowledge of the agent model’s architecture to obtain the gradient. These methods can be employed either in an untargeted manner, by maximizing the loss between the chosen action and itself, or in a targeted manner, by minimizing the loss between the chosen action and a specific target action.

- **FGSM** [31] is a fast computing method for crafting effective perturbed observations with $x' = x + \varepsilon \cdot a^{\varepsilon, X}$ with $a^{\varepsilon, X} = \text{sign}(\nabla_x L(f(x), y))$. In some case a variant is preferred with $a^{\varepsilon, X} = \nabla_x L(f(x), y)$.

- **BIM** [31] and **PGD** [32] are iterative versions of FGSM, BIM simply applies FGSM multiple times with small steps, while PGD is more refined and projects the adversarial example back into a feasible set after each iteration. They are more computation needing, since they are iterative methods that computes the gradient several times to craft more precise adversarial observations.

- **DeepFool** [33] is an iterative method. In each iteration, it linearizes the classifier’s decision boundary around the current input and then computes the perturbation to cross this linearized boundary.

- **C&W** [34] is a method that seeks to minimize the perturbation while ensuring that the perturbed input is classified as a specific target class.

These methods can be used on any agent having whether their type of action space (discrete or continuous).

- **JSMA** [35] is another gradient attack. It is more computationally expensive than FGSM, since it is an iterative method that craft perturbation with several iteration of computation of a Jacobian matrix for each output. It applies perturbation pixel by pixel, this make it particularly suitable for L_0 bounded perturbations.

- **XSMA** [36], **VFGA** [37], are methods based on JSMA, improving its effectiveness. Theses methods have been applied to RL in untargeted way [23, 27] in various type of environments, and in targeted way [38].

A representation of the integration and application of Gradient Attacks in crafting observation perturbations within an RL framework is shown in Figure 8.

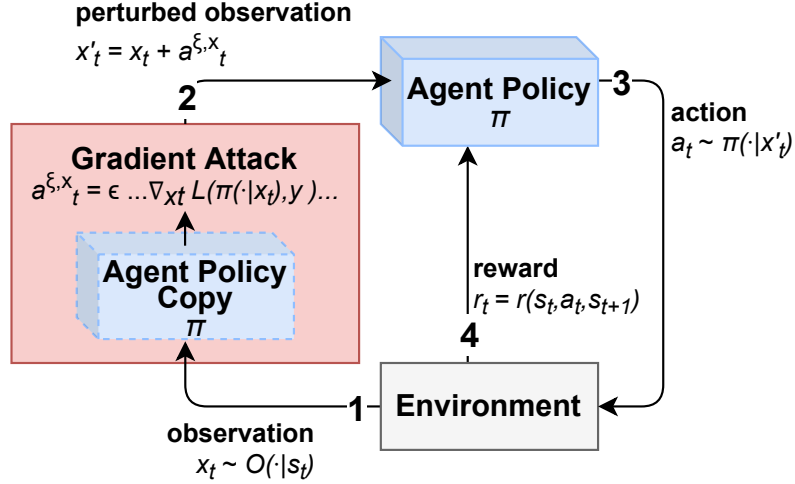


Fig. 8: Gradient Observation Attacks : The adversarial attack intercept the observation x_t , computes a perturbation $a_t^{\xi, X}$ by back-propagating the gradient of a loss in the neural network of a copy of the agent π , this perturbation is used to craft a perturbed observation x'_t , which is sent to the agent

Zeroth Order Optimization Methods (Black Box)

They represent an alternative category of direct adversarial attacks on observations. Unlike methods that rely on gradient computation, these attacks use optimization techniques that do not depend on gradient information to generate perturbations. Their primary objective is to alter the agent’s policy by producing a perturbed observation.

These methods employ various search techniques, such as random search, meta-heuristic optimization, or methods for estimating the gradient without direct computation. They operate in a black box setting, where the attacker has access only to the inputs and outputs of the model, without any internal knowledge of the agent.

Square Attack [39] is one such method that performs a random search within an ϵ -ball to discover adversarial examples. Although computationally demanding due to the number of iterations required for effective perturbation discovery, this method is applicable to any continuous observation space, including feature vectors, images, and signals, and to both discrete and continuous action spaces.

Finite Difference [28, 40] offers a technique for Gradient Estimation through querying the agent’s model, bypassing the need for white-box access. This estimated gradient is then utilized to craft a perturbed observation. This approach necessitates querying the neural network $2 \times N$ times for an input of size N .

A representation of the integration and application of Zeroth Order Optimization Methods in crafting observation perturbations within an RL framework is shown in Figure 9.

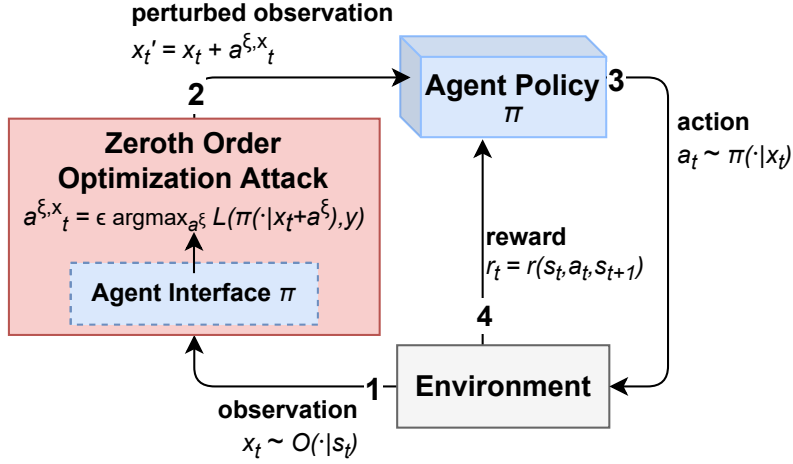


Fig. 9: Zeroth Order Optimization Observation Attacks : The adversarial attack intercept the observation x_t , computes a perturbation $a_t^{\xi, X}$ by querying the neural network of the agent π through an interface and applying a zeroth order optimization algorithm to maximize a loss, this perturbation is used to craft a perturbed observation x'_t , which is sent to the agent.

5.1.2 Minimize Reward

Several methods have been developed with the specific aim of producing perturbations in observations that directly minimize the reward obtained by an agent. These methods generate an adversarial action a^ξ that induces perturbations $a^{\xi, X}$ on the observation x' , thereby replacing the original observation x . As a result, the policy π yields a different output $\pi(x') \neq \pi(x)$, leading to undesirable situations characterized by lower rewards.

Most of these methods fall under the category of Adversarial Policy Learning Based Methods. As pointed out by [41], learning an optimal adversary to perturb observations is equivalent to learning an optimal policy in a new POMDP from its point of view as described in 4.2.1. The effectiveness of these approaches is largely due to their ability to leverage the sequential nature of the environment and the anticipation of future rewards, which aids in the development of effective Adversarial Policies. In contrast, Direct Optimization Methods are generally not used for this purpose, as they often struggle to capture the sequential dynamics of the environment and the implications for future rewards.

Adversarial Policy Learning Based Methods involve training an adversarial agent that initially learns to generate perturbations and subsequently uses this knowledge

to produce perturbations during inference. Typically, these methods employ an adversarial policy to generate perturbations $a^{\xi, X}$ from observations x , creating a perturbed observation x' . These methods require a training phase prior to being deployed as an attack, making them computationally intensive initially. However, once trained, these policies can be directly applied to generate perturbations with a significantly reduced computational cost when used in an attack scenario. In scenarios where the agent and the adversary are trained concurrently, these methods resemble techniques used in Generative Adversarial Networks (GAN). In this setup, the agent learns to perform its task while also becoming robust to the perturbations generated by the adversary. Simultaneously, the adversary refines its ability to create more effective perturbations to hinder the agent’s task performance. This co-learning approach enhances the adaptability and effectiveness of both the agent and the adversarial policy.

Classical Adversarial Policies (Black Box)

They are methods that employ an adversarial RL agent trained to create perturbations in observations. These methods function as black-box attacks, particularly during inference, and do not require comprehensive knowledge of the agent’s model to produce perturbations. Instead, their only requirement during the training phase is the ability to query the model for outputs based on various inputs. In the subsequent attack phase, the already-trained adversarial agent no longer needs additional information except for its own policy model parameters. To launch an attack, the agent simply performs a forward pass through its policy, generating a perturbation that results in a perturbed observation. The methods that follow this principle are Optimal Attack on Reinforcement Learning Policies **OARLP** [42, 43] and State Adversarial **SA-MDP** [44]. ATLA [45] also use this principle, but it is more focused on how to use it effectively in adversarial training, this is discussed further in Section 7.1. The adversary can use the same observation as the agent or augment its input with additional data, such as the agent’s action based on the original observation. This approach is highly flexible, allowing application across various observation spaces, including tabular data, feature vectors, images, and signals, and suitable for both discrete and continuous action spaces. Being black-box in nature, these methods only require the output of the agent model for a given input and do not need further information from the agent model. A representation of the integration and application of Adversarial Policies in crafting observation perturbations within an RL framework is shown in Figure 10.

Augmented Adversarial Policies (White Box)

Adversarial Policies can also be augmented with specific white-box techniques that can improve their performances to be more effective.

The Adversarial Transformer Network (**ATN**) method, as used and studied by [46, 47] shows that training an adversarial policy can also involve utilizing the gradients of the agent model. In this approach, the adversary is trained by back-propagating the agent’s loss through its input, which corresponds to the adversary’s output, and subsequently updating the adversary’s parameters. This technique effectively trains the adversary to generate perturbations that counteract the agent’s tendencies. During training, this method is considered white-box as it relies on the agent model’s

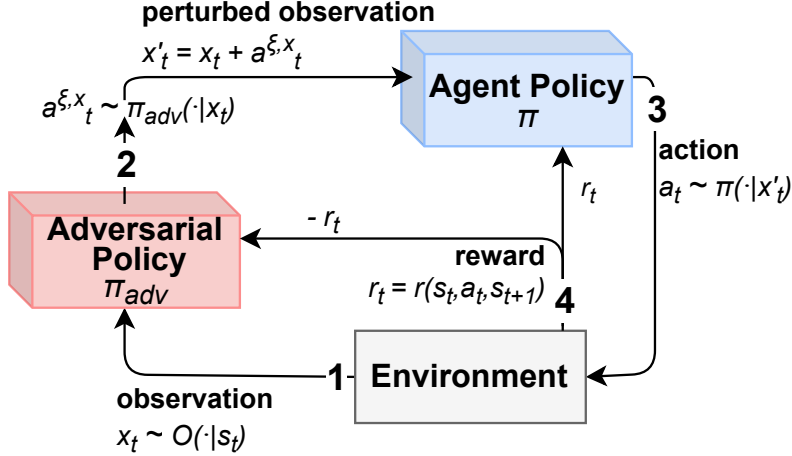


Fig. 10: Classical Adversarial Policy Observation Attacks : The adversarial policy intercept the observation x_t , computes a perturbation $a_t^{\xi, X}$ by a forward pass in its neural network, this perturbation is used to craft a perturbed observation x'_t , which is sent to the agent. The opposite reward as the agent is send to the adversarial policy to be trained.

gradients. However, at inference time, it functions as a black-box method since the trained policy alone is sufficient for operation. This methods are highly adaptable, applicable to any type of observation space, including tables, feature vectors, images, and signals, as well as to both discrete and continuous action spaces.

A representation of the integration and application of the Augmented Adversarial Policy **ATN** in crafting observation perturbations within an RL framework is shown in Figure 11.

Another augmented adversarial policy approach is **PA-AD** [48], this method craft an attack in two steps: First an RL based adversary, the Director adversary, is gives the direction of the perturbation wanted in the policy space, then this direction is given as target to the Actor adversary which is a direct Optimization method that compute the perturbation in the observation space to produce in order to make the agent choose the action wanted by the adversary. The director adversary in trained by RL by getting the opposite reward as the agent, thus improving its direction given to the actor adversary. The actor adversary cannot be learn, it only apply a direct optimization algorithm by optimizing the direction given by the actor. A representation of the integration and application of the Augmented Adversarial Policy **PA-AD** in crafting observation perturbations within an RL framework is shown in Figure 12.

5.2 Dynamic Alteration

This section presents an analysis of Dynamics Alteration Attacks for RL agents, which are methods that alter the transition function of the POMDP, they are useful to simulate mismatch between the dynamics of a deployment environment compared to the

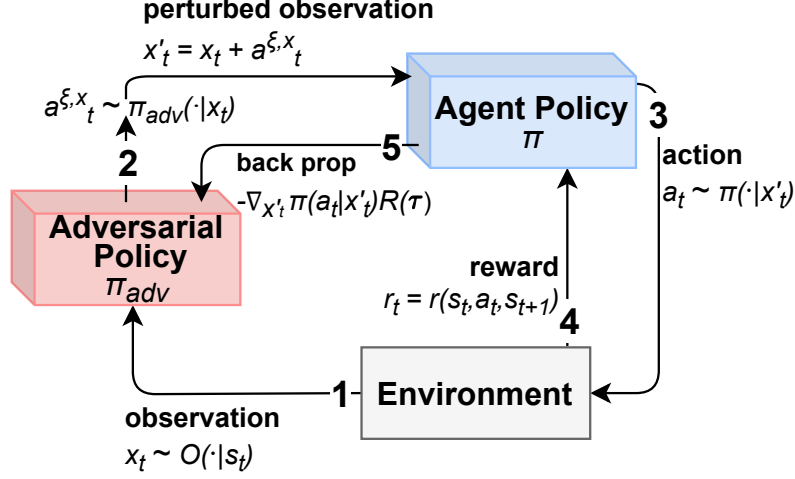


Fig. 11: Adversarial Transformer Network **ATN** Observation Attacks : The adversarial policy intercept the observation x_t , computes a perturbation $a_t^{\xi, X}$ by a forward pass in its neural network, this perturbation is used to craft a perturbed observation x'_t , which is sent to the agent. The opposite reward is sent to the agent, which back-propagate the loss to its input, then this loss is back-propagated in the adversarial policy network to be trained.

dynamics of the training environment, and they can be used to apply during the training of the agent to improve its robustness to unpredictable changes in the dynamics of the environment. Their goal is to produce an alteration of the transition function by producing a perturbation a^ξ at a certain state t with the current state being s_t . This perturbation a^ξ applied to any element of the transition function will have the consequence to lead to an alternative next state \tilde{s}_{t+1} , which is different than the original next s_{t+1} that would have been produced without alteration. To achieve this goal the attack can either :

- compute a perturbation $a^{\xi, S}$ to craft a perturbed state $\tilde{s}_t = s_t + a^{\xi, S}$.
- compute a perturbation $a^{\xi, A}$ to craft a perturbed action $a'_t = a + a^{\xi, A}$.
- compute a perturbation $a^{\xi, T}$ to directly alter the transition function T^Ω to induce alternative next state $T_\xi^\Omega(\tilde{s}_{t+1}|\dots, a^{\xi, T})$.
- compute a perturbation $a^{\xi, S+}$ to craft a perturbed next state $\tilde{s}_{t+1} = s_{t+1} + a^{\xi, S+}$.

As previously shown in Figures 5 7 4 6 in Section 4.2.2 All these methods have the consequence to lead to an alternative next state \tilde{s}_{t+1} . Theoretically, the amount of perturbation produced by such attacks should be measured as a distance between the original next state that would have been produced without alteration and the alternative next state given a norm L_p , $\|s_{t+1} - \tilde{s}_{t+1}\|_p$. But since it is often hard to redo several times the transition function of an environment, in practice often only the difference to the directly perturbed element is measured : either $\|s_t - \tilde{s}_t\|_p$, $\|a_t - a'_t\|_p$, $\|a^{\xi, T}\|_p$, or $\|\tilde{s}_{t+1} - s_{t+1}\|_p$ given the type of attack, thus making hard to compare disturbance magnitude between two different type of attacks that does not perturb the

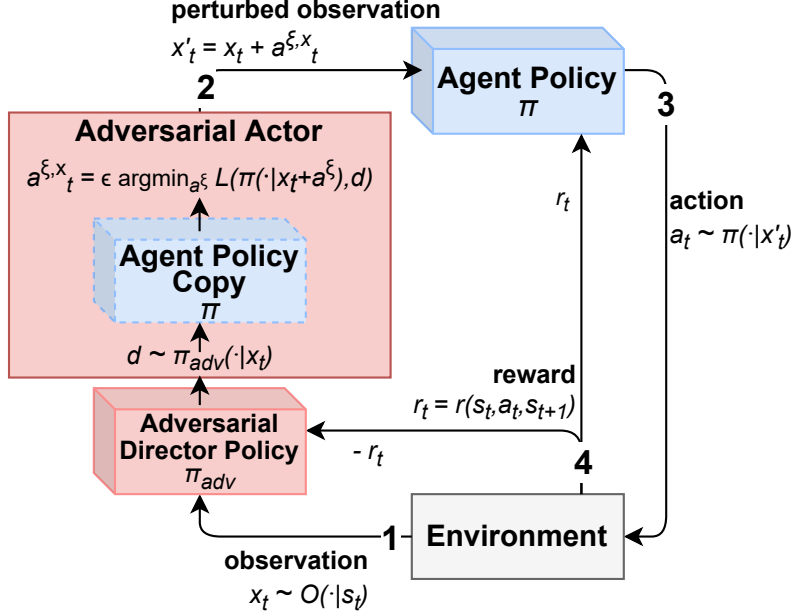


Fig. 12: PA-AD Observation Attacks : The adversarial policy intercept the observation x_t , the director adversary computes a direction in the policy space by forward pass in its neural network, the actor adversary computes a perturbation $a_t^{\xi, X}$ by direct optimization, this perturbation is used to craft a perturbed observation x'_t , which is sent to the agent. The opposite reward as the agent is send to the director adversary to be trained.

same element.

Tampering with the transition is a completely different approach than with the observation. The methods developed in this section assume that the environments simulate physical, real-life settings : the perturbations are more restricted, ruling out gradient-based methods, and their effects on the agent is now indirect. In this section we first discuss methods that are designed to minimize the rewards obtained by the agent by altering the transition, then we discuss methods that are designed to deviate the policy of the agent.

5.2.1 Minimize Reward

Certain methods are specifically designed to modify the dynamics of an environment, aiming to reduce the rewards an agent receives. These methods achieve this by generating an adversarial action a^ξ , which leads to an altered subsequent state \tilde{s}_{t+1} , diverging from the original next state s_{t+1} . As a result, the agent finds itself in an unfavorable situation \tilde{s}_{t+1} at the next step, potentially leading to reduced immediate or future rewards. Most of these methods fall under the category of Adversarial Policy

Learning Based Methods. This is because they are well-suited to exploit the sequential nature of the environment and the anticipation of future rewards, which aids in developing effective Adversarial Policies.

Adversarial Policies involve methods where an adversarial agent is trained to create perturbations. Initially, this agent learns how to generate these perturbations, and once trained, it can efficiently produce them during inference. Typically, these methods utilize an adversarial policy to create an adversarial action a^ξ , aimed at altering the environment’s transitions. The training of these methods must be completed before they are deployed for attacks, which makes the initial phase computationally demanding. However, once the training phase is over, the adversarial policy can be used directly to generate perturbations at a significantly reduced computational cost in attack scenarios. In such a setup, the primary agent learns to perform its task while also becoming resilient to the adversary’s perturbations. Concurrently, the adversarial agent refines its skills in creating more effective perturbations to hinder the primary agent’s task performance.

Classical Adversarial Policies (Black Box)

They utilize an adversarial RL agent trained to create adversarial actions that modify transitions within the environment. Functioning primarily as black-box attacks, especially during the inference phase, these methods do not necessitate comprehensive understanding of the agent’s model to produce perturbations. Instead, their main requirement during the training phase is the ability to query the model and obtain its outputs for various inputs. Once the adversarial agent completes its training, no additional information is needed beyond the parameters of its own policy model. During the attack phase, generating a perturbation involves simply executing a forward pass through the adversarial policy to create a perturbed observation. Examples of methods adhering to this approach include Robust Adversarial Reinforcement Learning (**RARL**) [49], along with its advancements such as Risk Averse (**RA-RARL**) [50], and Semi-Competitive (**SC-RARL**) [51]. These methods exemplify the application of Adversarial Policies in crafting effective black-box attacks in RL contexts. **FSP** [52, 53] also use this approach, but its points is more focused on how to use it effectively in adversarial training, this is discussed further in Section 7.1. These methods have been introduced as adding an adversarial action to an augmented version of the transition function T_ξ^Ω , a representation of the integration and usage of Adversarial Policies Attacks to add transition perturbations in an RL context is shown in Figure 13. The perturbation is added in the environment as previously shown in Figure 4. These methods can also be used to generate perturbations on the state s_t , the next state s_{t+1} , and also the action a_t as previously shown in Figures 5 6 7 in Section 4.2.2. The Probabilistic Action Robust **PR-MDP** method [54], follows the same principle as the other adversarial policy methods, the goal is to train an adversarial policy to generate perturbation, but specifically on the action space as previously shown in Figure 7. These methods are very versatile, they can be applied to any observation space (table, features vector, images, signals) and action space (discrete or continuous). When adding disturbances in the transition function directly or in the states there is the constraint of having activatable lever in the environment to be used by

the adversarial policy add disturbances, this constraint is not prevent when disturbing the action since the action itself is already the lever, but the attacks perturbing the action may have less means for disturbing the dynamics than methods that perturb state or transition. A representation of the integration and usage of Adversarial Policies Attacks to add transition perturbations in an RL context is shown in Figure 13.

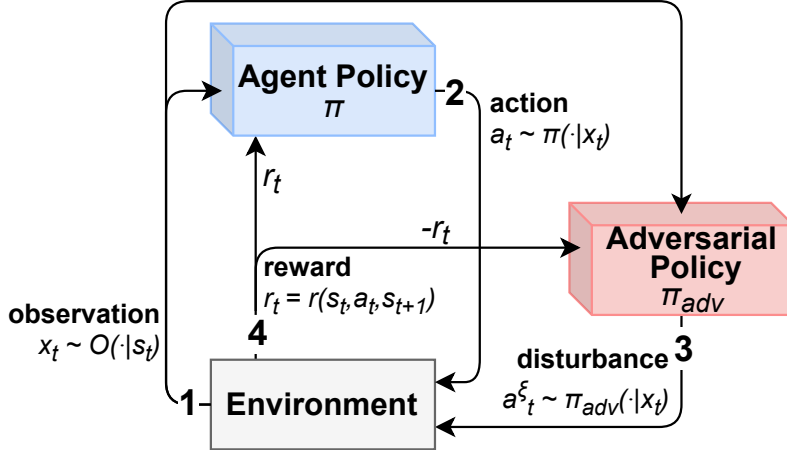


Fig. 13: Classical Adversarial Policy Dynamic Attack : The agent and the adversary get an observation x_t of the environment. The agent chooses the action a_t to apply and the adversary chooses the perturbation a_t^ξ to apply to alter the dynamics. The step function of the environment is run incorporating both agent action a_t and adversarial action a_t^ξ . The reward opposite to that of the agent is sent to the adversarial policy to be trained.

Other works like **APDRL** [55], **A-MCTS** [56], **APT** [57] and **ICMCTS-BR** [58] have used the concept of adversarial policy or adversarial agent, but more in the context of a real two players game, where an agent learns to do a task and an adversarial agent learns to make the agent fail. The key difference with methods previously discussed like RARL, is that here the environment itself is a two players game. The agent and the adversary are always here, contrary to methods previously discussed like RARL where the original setup is an agent alone learning to do a task and the adversary comes to challenge the agent and make it improve its performances for itself. So the methods presented in these works are very similar to RARL with some more specificities, and even if there were not presented in the context of the robustness for a single agent they can also be adapted and used in this setup.

Augmented Adversarial Policies (White Box)

Another possibility is to augment the information observed by the adversary with internal state of the agent like latent spaces or others. This is done by White-Box

Adversarial Policy **WB-AP** [59] which is very similar to RARL except that the adversary has white box access to the agent internal data. This enables to improve the attack effectiveness of the perturbations since the adversary can learn to adapt the perturbations to the internal states of the agent that is attacked. This method is white box since it requires access to the internal state of the agent. It can be applied on any observation, and action spaces. A representation of the integration and usage of Adversarial Policies Attacks to add transition perturbations in an RL context is shown in Figure 14

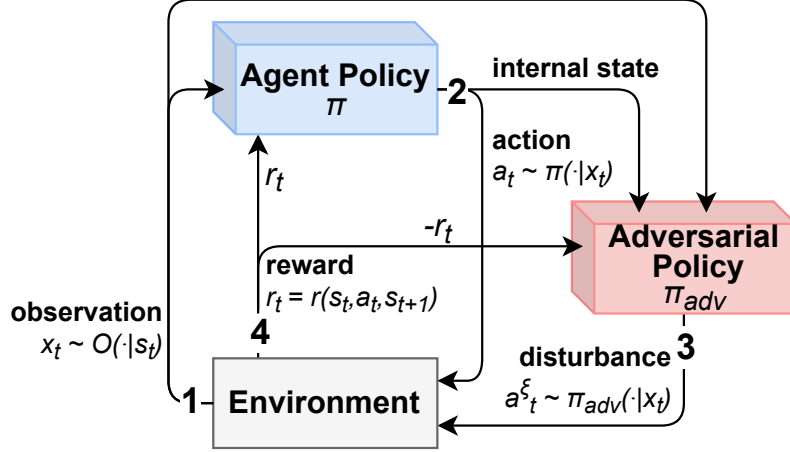


Fig. 14: White Box Adversarial Policy Dynamic Attack : The agent get an observation x_t of the environment and chooses the action a_t to apply. The adversary get the observation and some white box internal state of the agent and chooses the perturbation a_t^ξ to apply to alter the dynamics. The step function of the environment is run incorporating both agent action a_t and adversarial action a_t^ξ . The reward opposite to that of the agent is sent to the adversarial policy to be trained.

Direct Optimization Methods

They use direct optimization to generate adversarial action a^ξ to alter the transition of the environment. These methods on the contrary to Adversarial Policies does not require a prior training before usage in attacks. Although, they can be more computation needing when used in attacks since they have to solve an optimization problem at each perturbation rather than doing a forward pass in a neural network of an adversary policy.

First Order Optimization Methods: Gradient Attacks (White Box)

MAS, LAS [60, 61] are methods that alter the action a of the agent with a perturbation $a^{\xi, A}$ to make a perturbed action $a' = a + a^{\xi, A}$ that is less effective than the original action a . This method work only on RL agent that works with an actor network that

produce the action and a Q-critic network that estimate the Q-value of the observation-action tuple. The method is to apply a gradient attack on the Q-Critic network by computing the gradient of the Q-value with respect to the action $a_t^{\xi,A} = \epsilon \nabla_a Q(x_t, a_t)$ to minimize the Q-value estimated by perturbing the action. LAS is an extension of the MAS method which computes perturbation to apply over a sequence of future states to improve the long term impact of the attacks. This requires specific conditions to be applied such as an environment with repayable sequences, because the agent must be able to be reset at the specific state after the next pre-computed steps used to craft the perturbation. The scheme of the integration and usage of MAS attack in an RL context is shown in Figure 15.

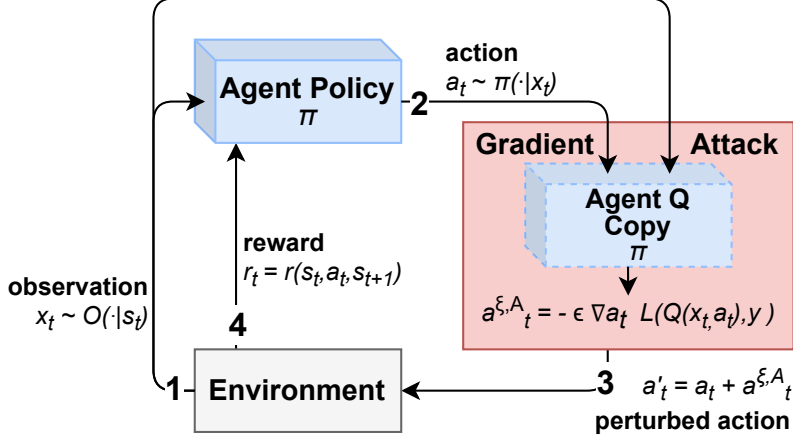


Fig. 15: MAS Dynamic Attacks on the Action: The agent and the adversarial attack get the observation x_t of the environment. The agent chooses the action a_t to apply, the adversarial attack computes the gradient of a copy of the Q-Critic Network of the agent with respect to the action a_t , this gradient is used as disturbance $a_t^{\xi,A}$ to add in the action $a'_t = a_t + a_t^{\xi,A}$. The step function of the environment is ran starting from the perturbed state \tilde{s}_t , with the agent action a_t .

Environment Attack based on the value-Critic Network **EACN** [62] is a method that apply a gradient attack to compute a perturbed observation x' and then use it to craft a perturbed state \tilde{s} . The general idea is to use the knowledge of the critic network's gradient to progressively increase the complexity of the task during training. At step t with state s and observation $x = O(s)$, EACN computes the gradient of the input of the Value-Critic Network V to minimize the output value. The value-Critic Network evaluates the Value function of the environment given the policy of the agent. So the method generates a perturbed input x' with $V(x') < V(x)$, this perturbed input is then used to create a perturbed state \tilde{s} with the property $x' = O(\tilde{s})$. Then the value estimated by the agent of the state is less than what it would have been without perturbation $V(O(\tilde{s})) < V(O(s))$. This attack can either be applied before or after the transition function. The method requires a Value-Critic Network as in the

PPO algorithm, but any other RL approaches can be considered by just adding the training of a Value-Critic Network on the resulting policy.

The main advantage of EACN is that it avoids training an adversarial policy. But it is less versatile than an adversarial policy since it needs one-to-one correspondence between the observations and the disturbances available, since the disturbances are computed from the gradient on the observations. So most of the time EACN is applicable only on environments where the observation space is a feature space, with modifiable features. The method works with an agent with any action space (discrete or continuous). A scheme of the integration and usage of EACN Gradient method on dynamic attacks in an RL context is shown in Figure 16.

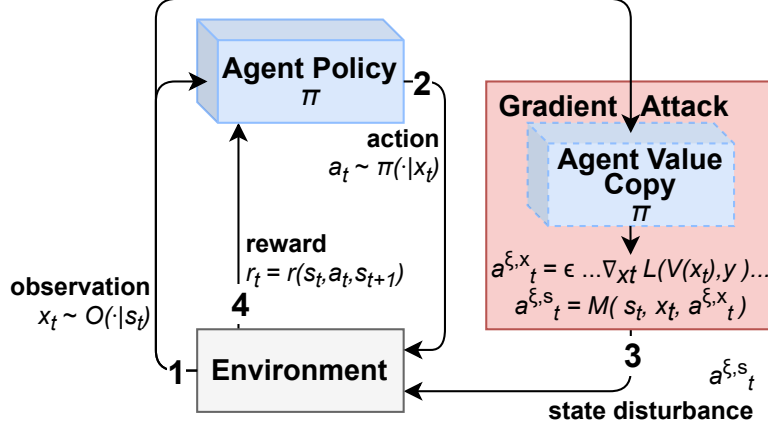


Fig. 16: Environment Attack based on the Critic Network **EACN** Dynamic Attacks : The agent and the adversarial attack get the observation x_t of the environment. The agent chooses the action a_t to apply and while the adversarial attack computes the gradient of a loss in a copy of the Value-Critic Network of the agent, this gradient is used as disturbance $a_t^{\xi, S}$ to add in the state $\tilde{s}_t = s_t + a_t^{\xi, S}$. The step function of the environment is ran starting from the perturbed state \tilde{s}_t , with the agent action a_t .

Some other approaches have been proposed like to generate perturbations in grid-world environments. These methods are WBA [63] which works by analyzing the Q table, and CDG [64] which works by analyzing the gradient of the Q network on the grid. But these methods add new obstacles in the grid, then changing the states-set. But our definition of perturbation of environments relies on dynamic alterations, by perturbing the transition function which is not the case for these methods.

5.2.2 Deviate Policy

Certain methods are designed to alter the transitions and to produce perturbation optimized to deviate the policy of the agent. By generating an adversarial action a_t^{ξ} that will induce an alternative next state \tilde{s}_{t+1} to replace the original next state s_{t+1} .

The agent will be placed at the next step in a situation \tilde{s}_{t+1} where it should change the action that it would initially have chosen.

First Order Optimization Methods: Gradient Attacks (White Box)

Environment Attack based on the Actor Network **EAAN** [62]. At step t with state s and observation $x = O(s)$, EAAN is used to compute the gradient of a loss on the input of the model of the agent to generate a perturbed input x' with $\pi(x') \neq \pi(x)$, this perturbed input is then used to create a perturbed state \tilde{s} with the property $x' = O(\tilde{s})$. Then, the action chosen by the agent at this step is different than what it would have been without perturbation $\pi(O(\tilde{s})) \neq \pi(O(s))$. This attack can either be applied before or after the transition function. EACN, this method is a bit less versatile than with an adversarial policy, since in addition to have activatable lever in the environment to be used to add disturbances, these lever need to have one-to-one correspondence with the observations used as inputs of the agent. So most of the time EAAN is applicable only on environments where the observation space is a feature space, with modifiable features. The method works with any action space (discrete or continuous). The scheme of the integration and usage of EAAN Gradient method on dynamic attacks in an RL context is shown exactly the same as for EACN which is shown in Figure 16, the only difference is that for EAAN the gradient is computed on the actor network π , rather than on the value-critic network V .

5.2.3 Other Objective

Classical Adversarial Policy Learning Based Methods

Environment-Search Attack **ESA** : [28] train an adversary to perturb the environment transition model M with an adversarial reward based on the distance between the disturbed state and the original state: the goal is to make small changes to M but to induce a completely different disturbed state. The scheme of the integration and usage of ESA on dynamic attacks in an RL context is shown exactly the same as for Adversarial Policy which is shown in Figure 13, the only difference is that in ESA reward get by the adversary is not the opposite of the reward of the agent, but is a reward based on the state distance.

6 Strategies of Attacks

Adversarial attack can be used in many different ways and there can be various approaches to use them. Often adversarial attack are introduced with a specific application strategy. In this section we develop about how various strategies are applied to use adversarial attacks.

6.1 Strategies for using White Box Attacks in Black Box Scenarios

Some strategies leverage white box attack methods in scenarios where the adversary has limited knowledge about the target agent (black box scenarios). Techniques in this category, like imitation learning, often involve understanding or approximating the

Component Alteration	Objective	Category	Perturbed Element	Model Knowledge	Method	
Observations Alteration see 5.1	Deviate Policy	Gradient Attack see 5.1.1	observation x	white-box	FGSM [65] BIM [31] PGD [32] DeepFool [33] C&W [34] JSMA [35] XSMA [36] VFGA [37]	
					Zeroth Order Optimization Attack see 5.1.1	observation x
	Minimize Reward	Adversarial Policy see 5.1.2	observation x	black-box	OARLP [42, 43] SA-MDP [44]	
				white-box	ATN [46, 47] PA-AD [48]	
Dynamics Alteration see 5.2	Minimize Reward	Adversarial Policy see 5.2.1	transition T^{Ω} state s or action a	black-box	RARL [49] RA-RARL [50] SC-RARL [51] A-MCTS [56] APT [57] ICMCTS-BR [58]	
				white-box	WB-AP [59]	
			action a	black-box	PR-MDP [54]	
			Gradient Attack see 5.2.1	action a	white-box	MAS, LAS [60, 61]
				state s	white-box	EACN [62]
			Deviate Policy	Gradient Attack see 5.2.2	state s	white-box
Other Objective	Adversarial Policy see 5.2.3	state s	black-box	ESA [28]		

Table 1: Characterization and Constraints of Adversarial Attack Methods for Reinforcement Learning : Summary of the Content of Section 5

internal mechanisms of the target system without complete access to its architecture or training data. This approach is particularly challenging as it requires the adversary to make accurate guesses or approximations about the target’s behavior based on limited observable outputs or effects. The **AEPI** strategy [66] targets black-box environments where the attacker has limited knowledge about the victim’s system. It uses Deep Q-Learning from Demonstration (DQfD) to imitate the victim’s Q-function, to the apply white box adversarial attacks. The **RS** strategy [44] is designed to not relying on the

victim’s Q-function’s accuracy. RS focuses on learning the Q-function of the victim’s policy and then applying any white box adversarial attacks.

6.2 Strategies for Timing and Stealthiness

Some Strategies are characterized by their focus on the timing of attacks and maintaining stealthiness. These strategies are designed to minimize detection while maximizing impact, often by carefully choosing when to launch an attack or by subtly altering agent behavior. This approach is particularly relevant in scenarios where avoiding detection is crucial, either for the success of the attack or to study the system’s vulnerabilities without triggering alarms.

K&S [67], and Strategically-Timed Attack **STA** [68], both concentrate on the timing of perturbations. [67] found that altering the frequency of FGSM perturbation injections can maintain effectiveness while reducing computational costs. Similarly, STA employs a preference metric to determine the optimal moments for launching perturbations, targeting only 25% of the states.

Weighted Majority Algorithm **WMA** [69] and Critical Point and Antagonist Attack **CPA and AA** [70]. Take the concept further by introducing more sophisticated timing strategies. WMA uses real-time calculations to select the most sensitive timeframes for attacks, while CPA and AA focus on identifying critical moments for injections, predicting the next environment state or using an adversarial agent to decide the timing.

Static Reward Impact Maps **SRIMA** [71] and Minimalistic Attack **MA** [72] emphasize efficiency and stealth. SRIMA method selects only the most influential features for perturbation, reducing the gradient computation cost, and is suitable for time-constrained environments. MA pushes stealth to the limits by altering only a small fraction of state features and timeframes, making it hard for the victim to detect the attack.

The Enchanting Attack **EA** [68] is a strategy that use a model that predict the future state depending on the action applied, and then based on the possible future states, apply an adversarial attack targeting the action that lead to the target state.

6.3 Strategies for Real Time and Resource Constraint Scenarios

Some strategies are optimized for efficiency and speed, making them suitable for real-time applications or scenarios with significant resource constraints. Universal attacks, which create perturbations that are broadly effective across different inputs or states, are a key part of this category. These techniques are valuable in environments where the computational power is limited or where decisions need to be made swiftly, such as in certain gaming or real-time decision-making applications.

Among the input-space adversarial examples, universal methods are particularly interesting when dealing with time-constrained environments: state-independent perturbations are computed offline, thus allowing for online injections with very small delay. These perturbations are short-term by nature, but may lead to controlling the victim’s policy and are therefore usually combined with adversarial policies.

CopyCAT [73] use an additive mask δ_a is computed offline for each action a , maximizing the expected $\pi(a, o_t^k + \delta_a)$ over a set of pre-collected observations $(o_t^k)_{k,t}$. Once each mask is computed, the chosen perturbation can be applied to the last observation o_t to make the agent follow any target policy. The authors do not focus on how to compute such policy, but there existing techniques were described earlier in the survey [74]. Experiments are conducted on Atari games, and CopyCAT provides better result than the targeted version of FGSM.

Universal Adversarial Perturbations **UAP-S, UAP-O** [75] inspired from a known DL-based universal method, UAP [76]. The adversary first gathers a set of observed states D_{train} and sanitizes it, keeping only the ones having a critical influence on the episode. Then, the unique additive mask is computed using the UAP algorithm [76]. In the case of Atari games, where a state s_t consists in several consecutive observations o_{t-N+1}, \dots, o_t , the perturbation can be state-agnostic (UAP-S), i.e. unique for all states but different for each o_i within a state, or observation-agnostic (UAP-O), i.e. unique for all observations. Experiments show promising results for both methods (with an advantage for UAP-S) on DQN, PPO and A2C agents in three different Atari games. Authors show that the SA-MDP method [44] is not sufficient to mitigate these attacks, and introduce an effective detection technique called AD^3 .

DAP [74] decoupled adversarial policy attack is also universal, though the paper is more focused on the adversarial policy aspect. Here each additive mask $\delta_{a,a'}$ is specific to the victim’s initial action a as well as the targeted action a' . States are then classified into the right (a, a') category using both the victim’s policy and the decoupled policy, and the designated mask is added to the observation if the switch policy allows it.

Category	Strategies
White Box Attacks in Black Box Scenarios see 6.1	AEPI [66] RS [44]
Timing and Stealthiness see 6.2	K&S [67] STA [68] WMA [69] CPA, AA [70] SRIMA [71] MA [72] EA [68]
Real Time and Resource Constraint Scenarios Methods see 6.3	CopyCAT [73] UAP-S / O [75, 76] DAP [74]

Table 2: Strategies for Adversarial Attack : Summary of the Content of Section 6

7 Adversarial and Robust Training

7.1 Robustness Strategies with Adversarial Training

Adversarial training in RL is a technique aimed at improving the robustness of RL agents against adversarial attacks. The general principle of adversarial training involves repeatedly exposing the agent to adversarial examples or perturbations during its training phase. This process is akin to inoculating the agent against potential attacks it might encounter in the real world or more complex environments. This method is conceptually akin to the principles of robust control [77], where the focus is on ensuring that control systems maintain stability and performance despite uncertainties and external disturbances. In RL, adversarial training can take various forms, but the core idea is consistent: the RL agent is trained not only with normal experiences drawn from its interaction with the environment but also with experiences that have been modified by adversarial perturbations [10]. These perturbations are typically generated using methods akin to those used in adversarial attacks – for example, by altering the agent’s observations or by modifying the dynamics of the environment. By training in such an adversarially challenging environment, the RL agent learns to perform its task effectively even when faced with manipulated inputs or altered state transitions. This makes the agent more robust and less susceptible to potential adversarial manipulations post-deployment. As detailed in 3.2 the training process often involves a sort of min-max game, where one tries to minimize the maximum possible loss that an adversary can induce. This approach mirrors robust control’s emphasis on preparing systems to handle worst-case scenarios and uncertainties, such as variations in system dynamics or external noise. Adversarial training in RL can also be related to GANs in supervised learning, where models are trained to be robust against an adversary that tries to generate examples to fool the model. In the RL context, this approach helps the agent to not only optimize its policy for the given task but also to harden it against unexpected changes or adversarial strategies that might be encountered, thereby enhancing its overall performance and reliability. Any adversarial attack method can be used in adversarial training, certain are better than others to improve robustness of the agent. And specific strategies fro applying attacks can be used to better improve robustness.

In the following the describe the different adversarial training strategies that can be applied :

7.1.1 Initial Adversarial Training

It Consists on training an adversary against an agent until convergence, and then adversarially train the agent against this agent for it to become robust. This technique works with adversarial attacks based on adversarial policies as done with the method RARL [49].

7.1.2 Continuous Adversarial Training

It consists on training the adversary to converge. And then co-train the adversary and the agent simultaneously.

- This approach of adversarial training is the one used for attack techniques based Direct Optimization such agent gradient or zeroth order. Since the attacks are based on the policy of the agent, the attack automatically re-adapt to the new policy, for example gradient attacks automatically compute gradient on the new parameters of the agents policy.
- However, for attack techniques based on adversarial policy this techniques is less usable since it is hard to maintain the effectiveness of the adversarial policy since both agent and adversarial policies are learned simultaneously, one agent could learn faster than the other and the the whole processes can be less effective. So for adversarial policies the previous approaches of alternate training is more preferable.

7.1.3 Alternate Adversarial Training

It consists on alternately training an adversary against an agent until convergence, and then train the agent against this adversary until convergence. And repeat the loop convergence of both agent and adversary. This technique works well with adversarial attacks based on adversarial policies as done with the method **ATLA** [45].

7.1.4 Fictitious Self Play

FSP [52, 53] is a strategy for applying adversarial policies attacks during the adversarial training to maximize the gain in robustness of the agent. The adversarial training is done alternatively between the agent and the adversary each until convergence. The idea is to not always apply fully effective attacks during the training phases of the agent, but sometimes applying a random or average disturbance. This strategy enhance RARL [49] to improve its effectiveness for improving generalization of the policy adversarially trained.

7.2 Other Robustness Strategies

Leveraging adversarial training for more robust and reliable DRL algorithms is the most used defense against adversarial attacks, the variety of methods available fitting each specific use case. However, the issue raised by [78] remain: the high amount of possible adversarial inputs makes the design of a unique and adaptive defense method unlikely. Combining design-specific adversarial training methods with more classic defensive measures is therefore an interesting lead to protect DRL models from malicious behaviour, as countermeasures can be implemented at each stage of the models' construction process.

First, a defendant may focus on the design stage to choose a robust network architecture for the agent's policy. [79] works on the impact of network architecture to a model's robustness shows that within a same parameters budget, some architectural

configurations are more robust than others. In addition, the authors show that reducing the deeper layers capacity also improves robustness. Though their findings focus on supervised models, they could be extended to DRL. Robust architecture may also be achieved through defensive distillation [80], i.e. training a smaller student network to imitate a larger teacher network with class probability outputs, resulting in a lighter network with more regularization. Distillation was shown to be effective for DRL [81], and [82] studied the relevance of distillation approaches according to different contexts. Observation alterations automatically transform any MDP into a POMDP, since the observation is not anymore deterministic, it has been shown that for POMDP, recurrent architecture improve performances of the policies [83], so do defend against observation alteration recurrent policies can be useful [45].

For improving robustness, the defendant may also improve regularization through noisy training rewards [84, 85]. Or increase exploration with noisy actions [30].

Worst-Case-Aware Robust Reinforcement Learning WocaR-RL [86] is a method that improve robustness to observations perturbation without attacking. It only works for adversarial attacks on the observations, it consist of constructing a ϵ -ball around the observation and computing the upper and lower bound of the action \hat{A} of the agent by convex relaxation of neural network. The the worst-attack action value network $\underline{Q}(x_t, a_t)$ is then learned based on $Q(x_t, \hat{a}_t)$, and then the agent is trained to maximize the worst-attack action value $\underline{Q}(x_t, a_t)$

Some other methods improve robustness defining a distribution of transition function for training the agent, the method **RAMU** [87] design a architecture enabling to learn policies across a distribution of transition function.

Finally, a classic method to add a layer of security during the testing stage is adversarial detection: identifying modified observations for denoising or removal. Detection may be done through successor representation [88], or using a separately trained model [89]. Adversarial detection is a widespread technique in supervised learning [90, 91], but it is limited when it comes to the RL context: indeed, environment-space perturbations produce legitimate (though unlikely) observations and are therefore very difficult to detect. In addition [92] showed that detection methods are usually not very versatile, and easily subject to small changes in the attack methods.

There is a wide spectrum of defense methods for DRL algorithms, each with their benefits and limitations. Combining several methods allows to cover different aspects of the adversarial threat, but the defendant must keep in mind that simply stacking defense layers does not necessarily improve robustness [93]: each method must be analyzed and selected with care according to the given context.

8 Discussion

8.1 Current issues

Adversarial methods for reinforcement learning is a recent yet booming research field. We encountered a few shortcomings in the tools and literature, the resolving of which would substantially help future usage and research.

8.1.1 Consistency

New methods to compute RL-based adversarial examples are published regularly, and the number and variety of existing malicious or defensive techniques is constantly increasing. Each paper addresses a specific aspect of adversarial RL, however the positioning of new methods amongst the pre-existing techniques is often unclear, and a more global vision is usually missing. We aim to address this need of a generic framework for classification, by introducing a precise and clear taxonomy and mapping existing methods onto it. Following a similar structure for presenting future work would highly improve the field’s coherence and clarity. This also applies to the metrics used in the experiments and to evaluate.

8.1.2 Code availability

A key element for the ongoing research is reproducibility. Indeed, a method’s validity relies heavily on its comparison with state-of-the-art techniques. Yet as some articles do not provide the corresponding code to replicate their experiments, reproducing existing methods becomes a tedious process for researchers, with sometimes unreliable results. We therefore wish to highlight the importance of publicly available code to make the best use of published works and progress further into the field.

8.1.3 Common tools and methodology

Comparing the performance of the various existing methods is challenging. The metrics used in the articles may differ depending on the authors’ objective, threat model, and chosen RL environments. Even if the metrics are similar, the results are also dependent on the chosen RL algorithm and its hyperparameters. To properly assess a methods efficiency, a standard methodology is needed: [94] propose such approach for RL robustness evaluation, and a few other usage-specific methodologies can be found [95, 96]. From a broader perspective, authors can follow a simple rule of thumb: using open source and peer-reviewed toolkits that provide reliable implementations of RL algorithms (Stable-baselines3 [97], Tianshou [98]), attacks methods (DRL-oriented [99] or not [100]), and robust training algorithms [101].

8.2 Open challenges

This survey is also an opportunity to highlight the major research directions to be explored in the future.

8.2.1 Explainability

The question of robustness and vulnerability to adversarial behaviour in RL boils down to the issue of trustworthy artificial intelligence. Explainable reinforcement learning (XLR) techniques are developed in order to deeply understand the RL decision-making process and improve its transparency [102, 103]. Recent works focus on the correlation between explainability and robustness for machine learning in general [104], and extending these concept to DRL may lead to significant progress in both domains.

8.2.2 Attack feasibility

Another interesting lead for the robustness of DL algorithms, and particularly for DRL, is the study of the practicality of adversarial examples. It is now acknowledged that an all-mighty, omniscient adversary can fool about any model: however, RL environments often model real-life situations with physical constraints. Recent works on the compatibility of adversarial examples with such constraint show promising results in the defense of DL models [105]. A thorough study on RL-based adversarial methods' feasibility could help improve the challenge of universally robust models.

9 Conclusion

This survey has provided an extensive examination of the robustness challenges in RL and the various adversarial training methods aimed at enhancing this robustness. Our work highlighted the susceptibility of RL agents to dynamic and observation alterations with adversarial attacks, underscoring a significant gap in their application in real-world scenarios where reliability and safety are paramount.

We have presented a novel taxonomy of adversarial attacks, categorizing them based on their impact on the dynamics and observations within the RL environment. This classification system not only aids in understanding the nature of these attacks but also serves as a guide for researchers and practitioners in identifying appropriate adversarial training strategies tailored to specific types of vulnerabilities.

Our formalization of the robustness problem in RL, drawing from the principles of distributionally robust optimization for both observation and dynamic alterations, provides a foundational framework for future research. By considering the worst-case scenarios within a controlled uncertainty set, we can develop RL agents that are not only robust to known adversarial attacks but also equipped to handle unexpected variations in real-world environments.

The exploration of adversarial training strategies in this survey emphasizes the importance of simulating realistic adversarial conditions during the training phase. By doing so, RL agents can be better prepared for the complexities and uncertainties of real-world operations, leading to more reliable and effective performance.

In conclusion, while our work sheds light on the current state of adversarial methods in DRL and their role in enhancing agent robustness, it also opens the door for further exploration. Future research should focus on refining adversarial training techniques, exploring new forms of attacks, and expanding the taxonomy as the field evolves. Additionally, there is a need for developing more sophisticated models that can balance the trade-off between robustness and performance efficiency. As DRL continues to evolve, the pursuit of robust, reliable, and safe autonomous agents remains a critical objective, ensuring their applicability and trustworthiness in a wide range of real-world applications.

Acknowledgements. This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute within the Con fiance.ai program.

References

- [1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Hiedmiller, M., Fiedjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* (2015). Accessed 2023-02-13
- [2] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. *Nature* (2016). Accessed 2023-02-13
- [3] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* (2019). Accessed 2023-02-13
- [4] Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-End Training of Deep Visuomotor Policies. *arXiv* (2016). <http://arxiv.org/abs/1504.00702> Accessed 2023-02-13
- [5] Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P.: Deep Reinforcement Learning for Autonomous Driving: A Survey. *arXiv* (2021). <http://arxiv.org/abs/2002.00444> Accessed 2023-02-13
- [6] Zhang, D., Han, X., Deng, C.: Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems* (2018)
- [7] Höfer, S., Bekris, K., Handa, A., Gamboa, J.C., Golemo, F., Mozifian, M., Atkeson, C., Fox, D., Goldberg, K., Leonard, J., et al.: Perspectives on sim2real transfer for robotics: A summary of the r: Ss 2020 workshop. *arXiv preprint arXiv:2012.03806* (2020)
- [8] Collins, J., Howard, D., Leitner, J.: Quantifying the reality gap in robotic manipulation tasks. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 6706–6712 (2019). IEEE
- [9] Ilahi, I., Usama, M., Qadir, J., Janjua, M.U., Al-Fuqaha, A., Hoang, D.T., Niyato, D.: Challenges and countermeasures for adversarial attacks on deep

- reinforcement learning. *IEEE Transactions on Artificial Intelligence* (2022)
- [10] Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., Peters, J.: Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction* 4(1), 276–315 (2022)
 - [11] Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems* (2019). Accessed 2022-10-12
 - [12] Chen, T., Liu, J., Xiang, Y., Niu, W., Tong, E., Han, Z.: Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity* (2019). Accessed 2022-10-12
 - [13] Sutton, R.S., Barto, A.G.: Reinforcement Learning, Second Edition: An Introduction. MIT Press, ??? (1998)
 - [14] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* (1992). Accessed 2023-02-22
 - [15] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with Deep Reinforcement Learning. arXiv (2013). <http://arxiv.org/abs/1312.5602> Accessed 2022-11-10
 - [16] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
 - [17] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2018)
 - [18] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. arXiv (2017). <http://arxiv.org/abs/1707.06347> Accessed 2022-11-10
 - [19] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *International Conference on Machine Learning* (2018). PMLR
 - [20] Kuznetsov, A., Shvechikov, P., Grishin, A., Vetrov, D.: Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In: *International Conference on Machine Learning* (2020). PMLR
 - [21] Vassilev, A., Oprea, A., Fordyce, A., Andersen, H.: Adversarial machine learning: A taxonomy and terminology of attacks and mitigations (2024)

- [22] Dahmen-Lhuissier, S.: ETSI - Best Security Standards | ETSI Security Standards. <https://www.etsi.org/technologies/securing-artificial-intelligence> Accessed 2022-12-02
- [23] Huang, S., Papernot, N., Goodfellow, I., Duan, Y., Abbeel, P.: Adversarial Attacks on Neural Network Policies. arXiv (2017). <http://arxiv.org/abs/1702.02284> Accessed 2022-11-22
- [24] Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.D.: The security of machine learning. *Machine Learning* (2010). Accessed 2022-11-10
- [25] Brunke, L., Greeff, M., Hall, A.W., Yuan, Z., Zhou, S., Panerati, J., Schoellig, A.P.: Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems* **5**, 411–444 (2022)
- [26] Morimoto, J., Doya, K.: Robust reinforcement learning. *Neural computation* **17**(2), 335–359 (2005)
- [27] Behzadan, V., Munir, A.: Vulnerability of deep reinforcement learning to policy induction attacks. In: Perner, P. (ed.) *Machine Learning and Data Mining in Pattern Recognition*. Springer, ??? (2017)
- [28] Pan, X., Xiao, C., He, W., Yang, S., Peng, J., Sun, M., Yi, J., Yang, Z., Liu, M., Li, B., Song, D.: Characterizing Attacks on Deep Reinforcement Learning. arXiv (2022). <http://arxiv.org/abs/1907.09470> Accessed 2022-11-23
- [29] Rahimian, H., Mehrotra, S.: Distributionally robust optimization: A review. arXiv preprint arXiv:1908.05659 (2019)
- [30] Hollenstein, J., Auddy, S., Saveriano, M., Renaudo, E., Piater, J.: Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance. arXiv (2022). <http://arxiv.org/abs/2206.03787> Accessed 2023-01-23
- [31] Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv (2017). <http://arxiv.org/abs/1607.02533> Accessed 2022-12-05
- [32] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
- [33] Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582 (2016)
- [34] Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: *2017 IEEE Symposium on Security and Privacy (SP)* (2017)

- [35] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (2016)
- [36] Césaire, M., Schott, L., Hajri, H., Lamprier, S., Gallinari, P.: Stochastic sparse adversarial attacks. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1247–1254 (2021). IEEE
- [37] Hajri, H., Cesaire, M., Schott, L., Lamprier, S., Gallinari, P.: Neural adversarial attacks with random noises. *International Journal on Artificial Intelligence Tools* (2022)
- [38] Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., Chowdhary, G.: Robust Deep Reinforcement Learning with Adversarial Attacks. *arXiv* (2017). <http://arxiv.org/abs/1712.03632> Accessed 2022-11-15
- [39] Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: *European Conference on Computer Vision*, pp. 484–501 (2020). Springer
- [40] Bhagoji, A.N., He, W., Li, B., Song, D.: Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491* (2017)
- [41] Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., Hsieh, C.-J.: Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems* **33**, 21024–21037 (2020)
- [42] Russo, A., Proutiere, A.: Optimal Attacks on Reinforcement Learning Policies. *arXiv* (2019). <http://arxiv.org/abs/1907.13548> Accessed 2022-10-17
- [43] Russo, A., Proutiere, A.: Towards optimal attacks on reinforcement learning policies. In: 2021 American Control Conference (ACC) (2021)
- [44] Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., Hsieh, C.-J.: Robust deep reinforcement learning against adversarial perturbations on state observations. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., ??? (2020). <https://proceedings.neurips.cc/paper/2020/hash/f0eb6568ea114ba6e293f903c34d7488-Abstract.html> Accessed 2022-11-15
- [45] Zhang, H., Chen, H., Boning, D., Hsieh, C.-J.: Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452* (2021)
- [46] Tretschk, E., Oh, S.J., Fritz, M.: Sequential Attacks on Agents for Long-Term Adversarial Goals. *arXiv* (2018). <http://arxiv.org/abs/1805.12487> Accessed 2022-10-17

- [47] Baluja, S., Fischer, I.: Learning to attack: Adversarial transformation networks. Proceedings of the AAAI Conference on Artificial Intelligence (2018). Accessed 2022-10-17
- [48] Sun, Y., Zheng, R., Liang, Y., Huang, F.: Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. International Conference on Learning Representations (ICLR) (2022)
- [49] Pinto, L., Davidson, J., Sukthankar, R., Gupta, A.: Robust adversarial reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning. PMLR, ??? (2017). <https://proceedings.mlr.press/v70/pinto17a.html> Accessed 2022-10-18
- [50] Pan, X., Seita, D., Gao, Y., Canny, J.: Risk averse robust adversarial reinforcement learning. In: 2019 International Conference on Robotics and Automation (ICRA) (2019)
- [51] Ma, X., Driggs-Campbell, K., Kochenderfer, M.J.: Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning. In: 2018 IEEE Intelligent Vehicles Symposium (IV) (2018). IEEE
- [52] Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International Conference on Machine Learning (2015). PMLR
- [53] Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect-information games. arXiv preprint arXiv:1603.01121 (2016)
- [54] Tessler, C., Efroni, Y., Mannor, S.: Action robust reinforcement learning and applications in continuous control. In: Proceedings of the 36th International Conference on Machine Learning. PMLR, ??? (2019). <https://proceedings.mlr.press/v97/tessler19a.html> Accessed 2022-10-24
- [55] Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., Russell, S.: Adversarial Policies: Attacking Deep Reinforcement Learning. arXiv (2021). <http://arxiv.org/abs/1905.10615> Accessed 2022-11-09
- [56] Wang, T.T., Gleave, A., Belrose, N., Tseng, T., Miller, J., Dennis, M.D., Duan, Y., Pogrebnik, V., Levine, S., Russell, S.: Adversarial Policies Beat Professional-Level Go AIs. arXiv (2022). <http://arxiv.org/abs/2211.00241> Accessed 2022-11-23
- [57] Wu, X., Guo, W., Wei, H., Xing, X.: Adversarial policy training against deep reinforcement learning. In: 30th USENIX Security Symposium (USENIX Security 21) (2021)
- [58] Timbers, F., Bard, N., Lockhart, E., Lanctot, M., Schmid, M., Burch, N., Schrittwieser, J., Hubert, T., Bowling, M.: Approximate exploitability: learning a best

- response. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2022)
- [59] Casper, S., Killian, T., Kreiman, G., Hadfield-Menell, D.: Red teaming with mind reading: White-box adversarial policies against rl agents. arXiv preprint arXiv:2209.02167 (2022)
- [60] Lee, X.Y., Ghadai, S., Tan, K.L., Hegde, C., Sarkar, S.: Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 4577–4584 (2020)
- [61] Tan, K.L., Esfandiari, Y., Lee, X.Y., Sarkar, S., *et al.*: Robustifying reinforcement learning agents via action space adversarial training. In: 2020 American Control Conference (ACC), pp. 3959–3964 (2020). IEEE
- [62] Schott, L., Hajri, H., Lamprier, S.: Improving robustness of deep reinforcement learning agents: Environment attack based on the critic network. In: 2022 International Joint Conference on Neural Networks (IJCNN) (2022). <http://arxiv.org/abs/2104.03154> Accessed 2022-11-08
- [63] Bai, X., Niu, W., Liu, J., Gao, X., Xiang, Y., Liu, J.: Adversarial examples construction towards white-box q table variation in dqn pathfinding training. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC) (2018)
- [64] Chen, T., Niu, W., Xiang, Y., Bai, X., Liu, J., Han, Z., Li, G.: Gradient Band-based Adversarial Training for Generalized Attack Immunity of A3C Path Finding. arXiv (2018). <http://arxiv.org/abs/1807.06752> Accessed 2022-10-20
- [65] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples. arXiv (2015). <http://arxiv.org/abs/1412.6572> Accessed 2022-12-05
- [66] Behzadan, V., Hsu, W.: Adversarial Exploitation of Policy Imitation. arXiv (2019). <http://arxiv.org/abs/1906.01121> Accessed 2022-10-17
- [67] Kos, J., Song, D.: Delving into adversarial attacks on deep policies. arXiv (2017). <http://arxiv.org/abs/1705.06452> Accessed 2022-11-22
- [68] Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., Sun, M.: Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. arXiv (2019). <http://arxiv.org/abs/1703.06748> Accessed 2022-10-14
- [69] Yang, C.-H.H., Qi, J., Chen, P.-Y., Ouyang, Y., Hung, I.-T.D., Lee, C.-H., Ma, X.: Enhanced adversarial strategically-timed attacks against deep reinforcement learning. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2020). <http://arxiv.org/abs/2002>

- [70] Sun, J., Zhang, T., Xie, X., Ma, L., Zheng, Y., Chen, K., Liu, Y.: Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning. arXiv (2020). <http://arxiv.org/abs/2005.07099> Accessed 2022-11-22
- [71] Chan, P.P.K., Wang, Y., Yeung, D.S.: Adversarial attack against deep reinforcement learning with static reward impact map. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. Association for Computing Machinery, ??? (2020). <https://doi.org/10.1145/3320269.3384715> Accessed 2022-11-22
- [72] Qu, X., Sun, Z., Ong, Y.-S., Gupta, A., Wei, P.: Minimalistic attacks: How little it takes to fool deep reinforcement learning policies. IEEE Transactions on Cognitive and Developmental Systems (2021)
- [73] Hussenot, L., Geist, M., Pietquin, O.: CopyCAT: Taking Control of Neural Policies with Constant Attacks. arXiv (2020). <http://arxiv.org/abs/1905.12282> Accessed 2022-10-20
- [74] Mo, K., Tang, W., Li, J., Yuan, X.: Attacking deep reinforcement learning with decoupled adversarial policy. IEEE Transactions on Dependable and Secure Computing (2022)
- [75] Tekgul, B.G.A., Wang, S., Marchal, S., Asokan, N.: Real-time Adversarial Perturbations against Deep Reinforcement Learning Policies: Attacks and Defenses. arXiv (2022). <http://arxiv.org/abs/2106.08746> Accessed 2022-10-24
- [76] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. arXiv (2017). <http://arxiv.org/abs/1610.08401> Accessed 2023-01-12
- [77] Dorato, P.: A historical review of robust control. IEEE Control Systems Magazine **7**(2), 44–47 (1987)
- [78] Wu, D.J.: Is attacking machine learning easier than defending it? (2017). <http://cleverhans.io/security/privacy/ml/2017/02/15/why-attacking-machine-learning-is-easier-than-defending-it.html> Accessed 2023-01-23
- [79] Huang, H., Wang, Y., Erfani, S., Gu, Q., Bailey, J., Ma, X.: Exploring architectural ingredients of adversarially robust deep neural networks. In: Advances in Neural Information Processing Systems. Curran Associates, Inc., ??? (2021). <https://proceedings.neurips.cc/paper/2021/hash/2bd7f907b7f5b6bbd91822c0c7b835f6-Abstract.html> Accessed 2023-01-23
- [80] Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a Defense

- to Adversarial Perturbations against Deep Neural Networks. arXiv (2016). <http://arxiv.org/abs/1511.04508> Accessed 2023-02-14
- [81] Rusu, A.A., Colmenarejo, S.G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., Hadsell, R.: Policy Distillation. arXiv (2016). <http://arxiv.org/abs/1511.06295> Accessed 2023-01-23
- [82] Czarnecki, W.M., Pascanu, R., Osindero, S., Jayakumar, S., Swirszcz, G., Jaderberg, M.: Distilling policy distillation. In: Proceedings of the Twenty-Second International Conference on Artificial Intelligence And Statistics. PMLR, ??? (2019). <https://proceedings.mlr.press/v89/czarnecki19a.html> Accessed 2023-01-23
- [83] Wierstra, D., Foerster, A., Peters, J., Schmidhuber, J.: Solving deep memory pomdps with recurrent policy gradients. In: Artificial Neural Networks–ICANN 2007: 17th International Conference, Porto, Portugal, September 9–13, 2007, Proceedings, Part I 17, pp. 697–706 (2007). Springer
- [84] Kumar, A.: Enhancing performance of reinforcement learning models in the presence of noisy rewards. Thesis (April 2019). <https://repositories.lib.utexas.edu/handle/2152/75758> Accessed 2023-01-23
- [85] Wang, J., Liu, Y., Li, B.: Reinforcement Learning with Perturbed Rewards. arXiv (2020). <http://arxiv.org/abs/1810.01032> Accessed 2023-01-23
- [86] Liang, Y., Sun, Y., Zheng, R., Huang, F.: Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning. *Advances in Neural Information Processing Systems* **35**, 22547–22561 (2022)
- [87] Queeney, J., Benosman, M.: Risk-averse model uncertainty for distributionally robust safe reinforcement learning. arXiv preprint arXiv:2301.12593 (2023)
- [88] Lin, Y.-C., Liu, M.-Y., Sun, M., Huang, J.-B.: Detecting Adversarial Attacks on Neural Network Policies with Visual Foresight. arXiv (2017). <http://arxiv.org/abs/1710.00814> Accessed 2023-01-23
- [89] Hickling, T., Aouf, N., Spencer, P.: Robust Adversarial Attacks Detection based on Explainable Deep Reinforcement Learning For UAV Guidance and Planning. arXiv (2022). <http://arxiv.org/abs/2206.02670> Accessed 2023-01-23
- [90] Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On Detecting Adversarial Perturbations. arXiv (2017). <http://arxiv.org/abs/1702.04267> Accessed 2023-01-23
- [91] Pang, T., Du, C., Dong, Y., Zhu, J.: Towards robust detection of adversarial examples. In: Advances in Neural Information Processing Systems. Curran Associates, Inc., ??? (2018). <https://proceedings.neurips.cc/paper/2018/hash/>

- [92] Carlini, N., Wagner, D.: Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. arXiv (2017). <http://arxiv.org/abs/1705.07263> Accessed 2023-02-15
- [93] He, W., Wei, J., Chen, X., Carlini, N., Song, D.: Adversarial example defense: Ensembles of weak defenses are not strong. In: 11th USENIX Workshop on Offensive Technologies (WOOT 17) (2017)
- [94] Behzadan, V., Hsu, W.: RL-Based Method for Benchmarking the Adversarial Resilience and Robustness of Deep Reinforcement Learning Policies. arXiv (2019). <http://arxiv.org/abs/1906.01110> Accessed 2023-01-26
- [95] Behzadan, V., Munir, A.: Adversarial Reinforcement Learning Framework for Benchmarking Collision Avoidance Mechanisms in Autonomous Vehicles. arXiv (2018). <http://arxiv.org/abs/1806.01368> Accessed 2023-01-26
- [96] Behzadan, V., Hsu, W.: Sequential Triggers for Watermarking of Deep Reinforcement Learning Policies. arXiv (2019). <http://arxiv.org/abs/1906.01126> Accessed 2023-01-26
- [97] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research (2021). Accessed 2023-01-26
- [98] Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., Zhu, J.: Tianshou: a Highly Modularized Deep Reinforcement Learning Library. arXiv (2022). <http://arxiv.org/abs/2107.14171> Accessed 2023-01-26
- [99] Behzadan, V., Munir, A.: Whatever Does Not Kill Deep Reinforcement Learning, Makes It Stronger. arXiv (2017). <http://arxiv.org/abs/1712.09344> Accessed 2023-01-26
- [100] Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambarzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., Gierke, W., Dong, Y., Berthelot, D., Hendricks, P., Rauber, J., Long, R., McDaniel, P.: Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. arXiv (2018). <http://arxiv.org/abs/1610.00768> Accessed 2023-01-26
- [101] Yuan, Z., Hall, A.W., Zhou, S., Brunke, L., Greeff, M., Panerati, J., Schoellig, A.P.: safe-control-gym: a Unified Benchmark Suite for Safe Learning-based Control and Reinforcement Learning in Robotics. arXiv (2022). <http://arxiv.org/abs/2109.06325> Accessed 2023-01-26

- [102] Heuillet, A., Couthouis, F., Díaz-Rodríguez, N.: Explainability in deep reinforcement learning. *Knowledge-Based Systems* (2021). Accessed 2023-01-27
- [103] Vouros, G.A.: Explainable deep reinforcement learning: State of the art and challenges. *ACM Computing Surveys* (2022). Accessed 2023-01-27
- [104] Datta, A., Fredrikson, M., Leino, K., Lu, K., Sen, S., Wang, Z.: Machine learning explainability and robustness: Connected at the hip. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, ??? (2021). <https://doi.org/10.1145/3447548.3470806> Accessed 2023-01-27
- [105] Sun, L., Tan, M., Zhou, Z.: A survey of practical adversarial example attacks. *Cybersecurity* (2018). Accessed 2023-01-27