Applied Intelligence (2024) 54:2631–2647
https://doi.org/10.1007/s10489-024-05275-7

Clustering-based attack detection for adversarial reinforcement learning

Rubén Majadas[1] · **Javier García**[2] · **Fernando Fernández**[1]

Accepted: 2 January 2024 / Published online: 6 February 2024

Abstract
Detecting malicious attacks presents a major challenge in the field of reinforcement learning (RL), as such attacks can force the victim to perform abnormal actions, with potentially severe consequences. To mitigate these risks, current research focuses on the enhancement of RL algorithms with efficient detection mechanisms, especially for real-world applications. Adversarial attacks have the potential to alter the environmental dynamics of a Markov Decision Process (MDP) perceived by an RL agent. Leveraging these changes in dynamics, we propose a novel approach to detect attacks. Our contribution can be summarized in two main aspects. Firstly, we propose a novel formalization of the attack detection problem that entails analyzing modifications made by attacks to the transition and reward dynamics within the environment. This problem can be framed as a context change detection problem, where the goal is to identify the transition from a "free-of-attack" situation to an "under-attack" scenario. To solve this problem, we propose a groundbreaking "model-free" clustering-based countermeasure. This approach consists of two essential steps: first, partitioning the transition space into clusters, and then using this partitioning to identify changes in environmental dynamics caused by adversarial attacks. To assess the efficiency of our detection method, we performed experiments on four established RL domains (grid-world, mountain car, carpole, and acrobot) and subjected them to four advanced attack types. Uniform, Strategically-timed, Q-value, and Multi-objective. Our study proves that our technique has a high potential for perturbation detection, even in scenarios where attackers employ more sophisticated strategies.

Keywords Adversarial reinforcement learning · Adversarial attacks · Change-point detection · Clustering applications

1 Introduction

Reinforcement learning (RL) vulnerabilities to adversarial attacks are generally well known [1]. In a pertinent attack scenario, an adversary may attempt to mislead a deployed RL system during testing by manipulating attack samples. Adversarial attacks often involve slight perturbations to observations from the environment, resulting in behavioral changes that can lead to catastrophic consequences [2]. Such vulnerabilities pose a significant threat in real-world situations [3, 4], as they can cause autonomous vehicles to swerve into oncoming traffic [5]. These attacks can have an instant impact on environmental dynamics. For instance, in a scenario where the agent is not under attack, an action a in the state s results in the agent being in s', but in an *under-attack* scenario, the same action a in s leads the agent to a different state s'_δ. Consequently, the agent might perceive that it is in a different state than its real one. This paves the way for the application of detection methods capable of identifying the environmental alterations resulting from attacks. In fact, these modifications prompt us to propose that adversarial attack detection and context detection in RL are closely linked [6, 7]: any sudden alteration of the environment dynamics is a context change, and in an adversarial RL setting, such a change may mean that the agent goes from a *free-of-attack* context to a *under-attack* context. Therefore, detecting such changes as early as possible is mandatory

✉ Rubén Majadas
rmajadas@pa.uc3m.es

Javier García
franciscojavier.garcia.polo@usc.es

Fernando Fernández
ffernand@inf.uc3m.es

[1] Departamento de Informática, Universidad Carlos III de Madrid, Avda. de la Universidad, 30, Leganés 28911, Madrid, Spain

[2] Universidad Santiago de Compostela, Rúa Lope Gómez de Marzoa, Santiago de Compostela 15782, La Coruña, Spain

in safety-critical domains to prevent dangerous situations or performance degradation.

Motivated by a current lack of research, this paper aims to provide a novel perspective on the adversarial detection problem in RL. Specifically, we approach this problem from a sequential perspective, where attacks can be perceived as abrupt changes in the input. To address this issue, we propose the utilization of a clustering-based approach from sequential analysis cited in [8, 9] to develop a *model-free* countermeasure for RL that can effectively identify disturbances in the environment's dynamics. Compared to other methods [6, 10–12], the suggested detection technique captures the sequential nature of an MDP, does not need prior knowledge of the transition and reward dynamics, and is able to detect changes in the environmental dynamics in real time, in the trajectory generated by adversarial attacks, and conducts a tractable univariate analysis of the environmental dynamics in the search for abrupt changes. Our detection system consists of two phases. Firstly, our approach produces a set of clusters from the collected transitions in a *free-of-attack* scenario. Then, our system discriminates between perceived environmental transitions, categorizing them as anomalies or normal based on a distance metric and a predefined threshold. This threshold can be adjusted to optimize results based on the specific application domain. To exemplify this, we employ ROC curves, which enable the visualization of the ideal trade-off between true positive and false positive rates.

This paper is structured as follows: Section 2 provides a concise explanation of RL and adversarial attacks for better comprehension of subsequent content. Section 3 outlines existing research on adversarial RL and various defense methods. Section 4 introduces a novel approach to detecting attacks by framing the problem as a context detection issue. Section 5 describes our clustering-based detection approach, and Section 6 reports on the evaluation performed. Finally, Section 7 provides a summary of the main findings and future research directions.

2 Background

In this section, we will explain the concepts of reinforcement learning and adversarial attacks relevant to classification and RL systems.

2.1 Reinforcement learning

We examine RL tasks formalized by a Markov Decision Process (MDP) [13]. An MDP is a 4-tuple $\mathcal{M} = (S, A, T, R)$ where S is the set of states, A is a set of actions available in each state, R is the reward function $R : S \times A \rightarrow \Re$ that assigns a reward r to each state-action pair s, a, and T is the transition function $T : S \times A \times S \rightarrow [0, 1]$ where $T(s, a, s')$ denotes the probability of transitioning from state $s \in S$ to state $s' \in S$ after taking action $a \in A$. The objective is to acquire a control policy, $\pi(s) = a$, that specifies the action $a \in A$ to be executed in a given input state $s \in S$ for the purpose of optimizing the return $J(\pi)$ as indicated in (1):

$$J(\pi) = \sum_{k=0}^{K} \gamma^k r_k \tag{1}$$

The discount factor, γ, determines how much the agent prioritizes future rewards, with values ranging from 0 to 1 ($0 \leq \gamma \leq 1$). The variable r_k represents the immediate reward received at step k. The agent interacts with the environment and generates transitions in the form of $\tau = \langle s, a, s', r \rangle$, where s' denotes the state to which the agent transitions, $s \in S$ is the current state, $a \in A$ is the action taken, and r is the reward received. Assuming a state space S in n-dimensions ($S \subset \Re^n$) where every state is a vector $s = (s_0, s_1, ..., s_n)$, and an m-dimensional action space A ($A \subset \Re^m$) where each action is a vector $a = (a_0, a_1, ..., a_m)$, a transition tau can be restated as $\tau =< s_0, ..., s_n, a_0, ..., a_m, s'_0, ..., s'_n, r >$, where τ belongs to the subset of $\Re^{(2n+m+1)}$. We calculate the distance between two transitions, τ_i and τ_j, using the Euclidean distance denoted in (2):

$$d(\tau_i, \tau_j) = \sqrt{\sum_k (\tau_{i,k} - \tau_{j,k})^2} \tag{2}$$

In the transitions i and j, the k-th component is represented by $\tau_{i,k}$ and $\tau_{j,k}$, respectively.

2.2 Adversarial attacks

Adversarial attacks are the result of adversarial examples in supervised learning, which have small but deliberate feature perturbations that lead to false predictions by supervised classifiers. Adversarial examples are a worry in machine learning, specifically deep learning, as they highlight that even cutting-edge models can be vulnerable to small changes in their input data. Adversarial examples provoke doubts concerning the durability and dependability of machine learning models in practical applications. Equation (3) is a formal explanation of what an adversarial attack consists of. To explain it, let x denote an observation and f represent a classification system. One can construct an adversarial example for classifier f by solving the following optimization problem.

$$\min_{\delta} d(x, x + \delta) \qquad subject\ to\ f(x) \neq f(x + \delta) \tag{3}$$

The aim of this task is to determine the smallest possible perturbation that can alter the decision made by classifier f

in comparison to its decision based on the initial input. The success of the attack is determined by achieving this goal, using a similarity metric d. The optimization problem in (3) can be solved by finding the perturbation δ of an observation x that causes the classifier f to output an incorrect class $f(x) \neq f(x+\delta)$. The same principles apply to RL. An attacker can target a victim who adheres to policy π by manipulating the victim's observations of the environment, with the objective of inducing the victim to select non-preferred actions where $\pi(s) \neq \pi(s+\delta)$. This can lead to a decrease in the cumulative reward or cause severe consequences for not just the victim but also any neighboring systems or persons.

3 Related work

In this section, we present a thorough literature review of previous research related to our approach, divided into two parts. The first part covers documented attacks found in existing literature, followed by a detailed analysis of currently employed detection mechanisms.

3.1 Attacks

In the literature, three distinct attack targets have been identified for injecting Adversarial Examples into any RL algorithm. Figure 1 illustrates these targets, where each adversary corresponds to a unique attack. To clarify, the targets are:

1. **State perception.** A significant portion of academic works focuses on the manipulation of state perception during both training [14, 15] and testing [16, 17] in machine learning. The primary aim of these studies is to delay, perturb, or falsify the learning agent's perceived state s in the face of malicious attacks. For instance, Uniform strategy [16] implements an iterative attack that misleads the victim in each iteration. Crafting an adversarial example during each iteration makes this process computationally complex. Therefore, some research attempts to minimize the number of attacks performed. For example, an attack takes place when the Q value surpasses a predetermined threshold [14] or when a learned policy has a vulnerability found based on the difference between the most and least effective action [18]. Novel methodologies propose to execute attacks using a multi-objective function. which aims to maximize the impact on the victim's policy and minimize the number of assaults [19].
2. **Action selected.** The attackers can also aim towards driving the victim to undesired states by disrupting the action a executed by the learning agent [20]. To execute this type of attack, the attacker must be familiar with the path to the malicious states.
3. **Training reward.** Attacks that perturb the reward function exist. These attacks pertain to changes made to the reward produced by the environment in response to actions taken by an RL agent [21]. The objective is to modify the victim's policy, driving it toward undesired states.

While our method can address all three types of attacks by analyzing the complete trajectory of the agent, we focus on preventing the first attack on state perception, since the specific attacks under evaluation caused perturbations in the perceived state. In addition, state perception attacks are extensively studied in the adversarial field. To facilitate a thorough and comprehensive analysis of multiple attack strategies, focus is directed towards the behavioral patterns of the first opponent illustrated in Fig. 1.

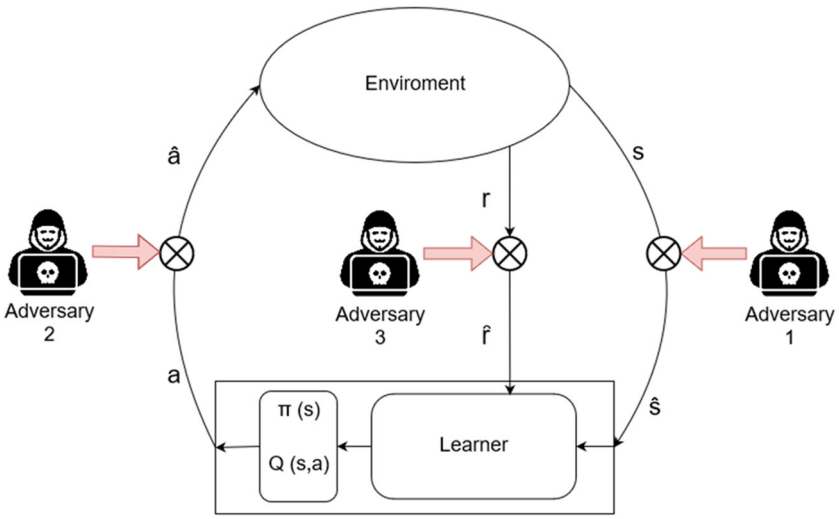


Fig. 1 RL attack targets. The adversary can compromise three different points of the learning process: the perceived state, the selected action, or the reward function

3.2 Detection mechanisms

Figure 1 can also be utilized to categorize existing defense methods since each one focuses on detecting these categories of attacks. However, many of the defense mechanisms employed in RL focus on identifying *adversarial observations*, neglecting the potential vulnerability of the actions and the reward signals, which are also susceptible to adversarial manipulation. These defenses often leverage techniques directly inherited from supervised learning paradigms. In this context, there are two main classes. The first category proposes training procedures to increase the robustness of a trained model against adversarial examples. Defensive distillation [22] and adversarial training [23] are methods to include in this category. Distillation is a process for transferring knowledge from different architectures; the goal of using several different architectures is to reduce computational complexity. This defensive distillation allows the authors to reduce the effectiveness of the adversarial examples. When defensive distillation is used, an eightfold increase in the number of features is required to disrupt the learning model [24].

In contrast, adversarial training defenses incorporate adversarial examples into the collected data to train an architecture that detects novel perturbations [4]. In this method, the learned model loses a small amount of accuracy in predicting clean examples, but adversarial training creates robustness to adversarial examples. Although distillation and adversarial training can achieve some success, they have two major drawbacks. First, training models that require attacks to learn to discriminate are not feasible in safety-critical systems where a single attack can result in catastrophic consequences. Additionally, the trained model can be easily fooled by adversarial examples generated by attack methods not encountered during training. Therefore, it is crucial to establish efficient detection methods that can be applied to diverse domains to counteract adversarial samples, even those that are novel to the system.

The second category focuses on detecting the statistical differences between the adversarial examples and the legitimate data [25]. For example, using a binary classifier to detect adversarial examples [26]. In this case, the authors show that a binary classifier trained with three different adversarial crafting methods is able to detect almost all attacks. However, they do not guarantee the detection of attacks not used in the training data. Other works based on statistical data have tried to extend their defenses against more adversarial crafting methods. Since these defenses are based on statistical data [11], they are not able to detect specific attacks. The collection of adversarial inputs must be large enough to generate a statistical bias capable of detecting the presence of unexpected behavior. Another work in this category develops a detector based on the Mahalanobis distance between the input variables [27]. The difference between the Euclidean distance and the Mahalanobis distance is that the latter takes into account the correlation between the variables. However, to detect attacks in an RL task, it is not only relevant whether an individual observation belongs to the training or test distribution or not. In an RL setting, it is extremely important to ensure the coherence of the transition and the reward the agent receives from the underlying MDP. Therefore, all these methods are effective in detecting single *adversarial observations*, but they forget the sequential nature of an RL task, where a sequence of states, actions, and rewards takes place. To sum up, they are blind to *adversarial transitions*. Such *adversarial transitions* may be composed of states belonging to the training or test distribution, which is even more unlikely for approaches that rely solely on observations to detect them [11, 25]. In contrast, we focus on the problem of detecting adversarial examples in sequential decision tasks. For this purpose, we exploit the coherence of the dynamics of the environment.

In this paper, we suggest that adversarial attacks can affect the distribution of transitions and rewards in RL environments. Similarly, such attacks affect the distribution of the training data in a supervised classification task [11]. Therefore, the abrupt perturbation of the environmental dynamics allows us to identify adversarial attacks. To the best of our knowledge, adversarial attack detection techniques have never been tackled from this novel perspective in RL. Some of these approaches are based on change-point detection, which involves a variation of the statistical features. And, it is relevant to keep in mind that this novel perspective opens the door to apply existing detection approaches also in an adversarial context [6], where they formalize an RL algorithm with context detection (RL-CD) to deal with non-stationary environments. Aiming to improve these results, other works focus on classifying the transitions into two categories: known and unknown, depending on a quality measure [7]. If this metric exceeds a certain threshold, it is considered a context change. However, RL-CD requires a set of parameters to be tuned according to the problem. Thus, this quality measure mainly depends on this ad hoc configuration. To solve this complex tuning task, another work proposes an incremental CUMSUM, building a library of models and policies for each type of context [28]. The limitation of this approach is the complexity of computing different policies and models. Furthermore, all of these approaches require *a priori* knowledge of the transition and reward dynamics, but this assumption rarely holds in the real world. In contrast, others detect the statistical differences between episodes, but these approaches are unable to detect statistical changes between time steps [12, 29, 30]. Such *episodic* detection techniques are useless in an adversarial context, where it is necessary to detect attacks as soon as possible.

In contrast to all previous approaches, this paper proposes a *model-free* countermeasure for the rapid detection of adversarial attacks based on the abrupt changes they produce in the sequence of transitions. Furthermore, this analysis is not performed from a multivariate point of view, but from a more compact and tractable univariate perspective. Such a multivariate analysis of transitions is not feasible due to the *curse of dimensionality*: the high dimensionality of RL tasks prevents us from using multivariate methods [10]. Consequently, a clustering-based approach is proposed in this paper to transform the analysis of the sequence of transitions from a multivariate to a univariate problem.

4 Problem formulation

Let $\mathcal{M} = (S, A, T, R)$ be an MDP in which an agent, which we will call the victim, has correctly learned a near-optimal policy π. Assume that the victim interacts with $\mathcal{M}$ using π in a *free-of-attack* context. At each time step, π produces experience tuples of the form $\tau = \langle s, a, s', r \rangle$ derived from T and R. Then there exists a time step $k > 0$ at which an adversary begins to perturb the states perceived by the victim, $s \rightarrow s_\delta$, with $s, s_\delta \in S$ and $s \neq s_\delta$. In this new context, using the same policy π, the victim will perceive experience tuples of the form $\tau = \langle s, a, s'_\delta, r \rangle$, where for each state-action pair $\langle s, a \rangle$ the next state s'_δ is not chosen according to T, but from a different distribution T_δ. Similar reasoning applies if the adversary attacks the actions or even the reward signal that the victim perceives from the environment. Thus, the adversary can perturb the functions T and R perceived by the victim, possibly at the same time. Hence, there exist time steps in which the trajectories sampled from the environment are associated with a new adversarial task $\mathcal{M}_\delta = (S, A, T_\delta, R_\delta)$ that the adversary induces in the victim whenever he attacks with $T_\delta \neq T$ and/or $R_\delta \neq R$. In this scenario with two MDPs, $\mathcal{M}$ and $\mathcal{M}_\delta$, the problem of attack detection is reduced to the problem of change-point detection, i.e., detecting the time steps at which the environment model changes. For example, at time k the environment model changes from e.g. $\mathcal{M}$ to $\mathcal{M}_\delta$, in which case the victim changes from a *free-of-attack* context $\mathcal{M}$ to a *under-of-attack* context $\mathcal{M}_\delta$. The context can also change from $\mathcal{M}_\delta$ to $\mathcal{M}$, in which case the victim will no longer receive attacks.

It is important to note that from a sequential analysis point of view, the problem of context detection reduces to detecting abrupt changes in streaming data. In RL, a data stream is a sequence of transitions as in (4):

$$\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_k, \tau_{k+1}, \tau_{k+2}, \tau_{k+3}, \ldots\} \tag{4}$$

where τ_t is the t-th transition perceived by the victim at time step t. The change point detection problem in the Γ sequence can be formulated in terms of testing the null hypothesis $\mathcal{H}_0$ against the alternative hypothesis $\mathcal{H}_1$ [31]. $\mathcal{H}_0$ asserts that at the current step t the model parameter remains the same, so the transition $\tau_t \in \Gamma$ was derived using $\mathcal{M}$. The alternative $\mathcal{H}_1$ claims that at the current step t the model parameters change and $\tau_t \in \Gamma$ is derived from $\mathcal{M}_\delta$. In an adversarial setting, we can obtain the expression of (5):

$$\begin{cases} \mathcal{H}_0 \neg attack \\ \mathcal{H}_1 \ attack \end{cases} \tag{5}$$

where the hypothesis is a logical expression *attack* that indicates whether an attack occurs. A powerful strategy for implementing the attack point detection mechanism depicted in (5) is to use a clustering-based approach as described in Section 5. This approach aims to detect attacks in $\mathcal{M}$ in a timely manner without introducing false negatives or positives.

5 Clustering-based attack detection

A clustering-based approach to detecting attacks offers two distinct benefits compared to other methods of detecting change-points [6, 7, 10, 28]. Firstly, it enables the development of a "model-free" attack detector that does not require approximation of the transition functions T or T_δ or the reward functions R or R_δ [6, 7, 28]. On the contrary, it converts a complex multivariate change-point detection problem into a simpler and more manageable univariate change-point problem, as explained in Section 5.1. Our recommended strategy consists of two stages. Firstly, we obtain knowledge on the transition space partition (Section 5.1). Secondly, we employ the learned partition to identify adversarial attacks (Section 5.2).

5.1 Clustering of the transition space

In the initial stage of the proposed approach, the aim is to develop a partition $\mathcal{C}$ of the transition space through the transitions obtained from a policy π in a *free-of-attack* context. The first step of the algorithm is represented by Algorithm 1, which requires inputs including the number of episodes, H, the number of steps, K, per episode, the policy, π, and the number of clusters, k.

Algorithm 1 stores the transitions $\tau = \langle s, a, s', r \rangle$ that the agent experiences with π in $\mathcal{T}$ (line 7). After H episodes, it creates a k-means model $\mathcal{C} = \{c_0, c_1, \ldots, c_k\}$ utilizing the transitions from $\mathcal{T}$ as the training set (line 9). Here, c_i represents the i-th centroid with $c_i = \langle s_i, a_i, s'_i, r_i \rangle$. It generates a list, denoted as $\beta = \{\beta_0, \beta_1, \ldots, \beta_k\}$, where each

Algorithm 1 First step: clustering construction.

Require: H, K, π, k
1: Set $\mathcal{T} = \emptyset$
2: **for** $h = 0$ to H **do**
3: Initialize s
4: **for** $n = 0$ to K **do**
5: Choose action a derived from $\pi(s)$
6: Take action a, observe s' and r
7: $\mathcal{T} := \mathcal{T} \cup \langle s, a, s', r \rangle$
8: $s \leftarrow s'$
9: **end for**
10: **end for**
11: $\mathcal{C}, \beta \leftarrow$ k-means(k,$\mathcal{T}$)
12: **return** $\mathcal{C}$, β, $\mathcal{T}$

Algorithm 2 Second step: context change detection.

Require: H, K, π, $\mathcal{C}$, β
1: $t \leftarrow 1$
2: **for** $h = 0$ to H **do**
3: Initialize s
4: **for** $n = 0$ to K **do**
5: Choose action a derived from $\pi(s)$
6: Take action a, observe s' and r
7: $\tau_t := \langle s, a, s', r \rangle$
8: $d_{\tau_t} \leftarrow \min_{0 \le j \le p} d(\tau_t, c_j)$
9: **if** $attack = [d_{\tau_t} > \beta_j]$ **then**
10: Stop execution
11: **end if**
12: $s \leftarrow s'$
13: $t \leftarrow t + 1$
14: **end for**
15: **end for**

β_i represents the radius of the i-th cluster, as computed in (6).

$$\beta_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d(\tau_j, c_i) \tag{6}$$

The algorithm uses the resulting partition of the transition space $\mathcal{C}$ and the list of thresholds β to detect adversarial attacks, where τ_j refers to the j-th transition in the i-th cluster, and n is the number of instances in that cluster (line 9).

It is worth noting that the sequence Γ depicted in (4) consists of multiple variable transitions $\tau \subset \Re^{(2\times n)+m-1}$, necessitating a multivariate analysis to detect change points. However, the partition $\mathcal{C}$ resulting from Algorithm 1 enables us to convert the multivariate analysis of Γ into a univariate analysis by simply considering the sequence $\Gamma^{\mathcal{C}}$ in (7).

$$\Gamma^{\mathcal{C}} = \{d_{\tau_1}, d_{\tau_2}, d_{\tau_3}, \ldots, d_{\tau_k}, d_{\tau_{k+1}}, d_{\tau_{k+2}}, d_{\tau_{k+3}}, \ldots\} \tag{7}$$

Here, $d_{\tau_t} \in \Re$ represents the Euclidean distance between the transition τ_t in the t-th time step and its nearest centroid $c_i \in \mathcal{C}$. This reduction enables detection of abrupt changes in the univariate sequence $\Gamma^{\mathcal{C}}$ for the attack detection problem, as elaborated in Section 5.2.

5.2 Detection of adversarial attacks

Roughly speaking, the $\mathcal{C}$ partition serves as a coarse representation of the trajectory followed by the agent in a *free-of-attack* scenario. Consequently, the $\Gamma^{\mathcal{C}}$ series can recognize unusual deviations in the agent's trajectory due to adversarial attacks in an *under-attack* scenario. The proposed detection mechanism is demonstrated in Algorithm 2.

Algorithm 2 takes as input the number of episodes H, the number of steps per episode K, the policy π, the transition space partition $\mathcal{C}$, and a list of thresholds β. After obtaining a suitable partition of the transition space $\mathcal{C}$ in the previous step, we calculate the distance d_{τ_t} between the perceived transition τ_t at time step t and its nearest centroid $c_j \in \mathcal{C}$ (line 8 in Algorithm 2). Then, Algorithm 2 compares this distance with the threshold β_j. Consequently, the logical expression $attack$ shown in (5) is reduced to $attack = [d_{\tau_t} > \beta_j]$ (line 9). If d_{τ_t} is greater than β_j, the transition is deemed an *adversarial transition*, meaning a context change has been deduced from an adversarial attack. This is a methodology for model fitting wherein a change is acknowledged if a new transition does not fit into any of the current clusters. If the system detects a context shift from 'free-of-attack' to 'under-attack,' the agent ceases its execution (line 10).

Figure 2 shows an example of how our detection system works. The figure displays transition points as blue and red markers. Upon creating the cluster model $\mathcal{C}$, we obtain a set of centroids $c_j \in \mathcal{C}$ (green stars), and establish thresholds β_j as the radius of each centroid. The radius (β_j) varies based on the training transitions associated with each cluster. Thus, instances that exceed the threshold distance to the nearest centroid are identified as normal transitions (blue markers), while the remaining instances (red dots) are classified as adversarial transitions.

It is crucial to recognize that the thresholds significantly impact the effectiveness of the suggested detection mechanism. In tasks involving security, these values cannot undergo adjustments in the deployment stage while anticipating the system to have received enough attacks to differentiate between those that are and are not an attack, because even a single attack can have disastrous consequences. Therefore, we have opted to calculate their values heuristically in the initial algorithm step, as outlined in (6). These computed values of β enable the Algorithm 2 to efficiently identify attacks at an early stage.

6 Evaluation

In this section, we present the results obtained by using our proposed detection mechanism in the grid world and

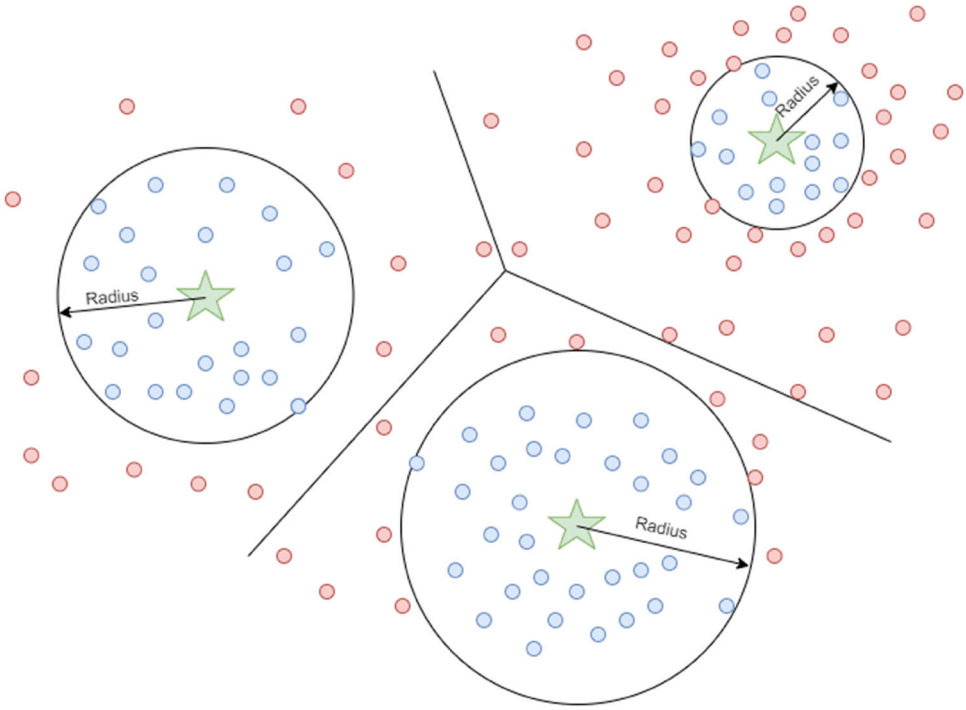


Fig. 2 Representation of the partition of the transition space

three well-known OpenAI games: carpole, mountain car, and acrobot. We design the experiments (i) to demonstrate that the transition-based detector proposed in this paper is able to capture the dynamics of RL tasks, in contrast to state-based detectors that focus only on single observations (Section 6.2); (ii) to validate the ability of the proposed approach to detect context changes from a *free-of-attack* to a *under-attack* context (Section 6.3); (iii) to validate the proposed approach from a binary classification perspective, which allows us to classify the transitions into two groups: adversarial and non-adversarial transitions (Section 6.4). Before doing so, we first present the experimental setting (Section 6.1).

6.1 Experimental setting

We evaluate our attack detection approach based on clustering in three well-established domains. Initially, we perform an initial proof of concept by utilizing a sample domain, such as a 10×10 grid. Subsequently, we refine our methodology in three sophisticated OpenAI domains, namely carpole, mountain car, and acrobot. In all scenarios, we presume that the victim had previously learned a policy π in a free-of-attack setting. For the purpose of this study, we implement a tabular Dyna-Q in the grid domain and utilize the DQN algorithm for the OpenAI environments. Table 1 displays the parameter values required for learning the policy π. Each domain is described in terms of its state space $\mathcal{S}$ and action space $\mathcal{A}$, the algorithm employed to learn the policy π, the learning rate α, and the maximum number of steps per episode K. We set the number of episodes to 1000 and the discount factor to 0.99 in all domains. It is important to emphasize that the domains employed in this study originate from the OpenAI Gym framework, a widely recognized platform for the development of Reinforcement Learning algorithms. Consequently, the specific configuration of the state space, action space, and the parameter K was not chosen by the authors, but rather, they adhere to the predefined settings within these Gym implementations. With respect to the "Algorithm" column in Table 1, it is essential to acknowledge that both the selection of the RL algorithm, network architectures, and the parameter denoted as α were informed by prior research endeavors that have demonstrated their effectiveness [19].

After π converges, the policy is used to generate the transition space. Later, the transition space will be used to build the partition ($\mathcal{C}$) and compute each radius's distance (β). The number of transitions $|\mathcal{T}|$ utilized by k-means in Algorithm 1 is comparable across the four domains. We conducted 200 test episodes to generate a similar number of instances. Combined, we have 25,000 transitions for each environment. We evaluated the number of clusters using the values k = 64, 256, 1024. Our observation shows that using more than 1024 clusters does not yield a significant improvement and only complicates the partition creation process in terms of memory and time.

Finally, we evaluate our detection approach against four cutting-edge strategies: the Uniform attack [16],' the

Table 1 Parameter setting of the learning process

Domain	$\mathcal{S}$	$\mathcal{A}$	Algorithm	α	K
Grid World	(x, y) $x, y \in [1, 10]$	$< 0, 1, 2, 3 >$	Dyna-Q	10^{-1}	30
Mountain car	(p, v) $p \in [-1.2, 0.6]$ $v \in [-0.07, 0.07]$	$< 0, 1, 2 >$	DQN $(2 \times 24 \times 24 \times 3)$	10^{-3}	200
Carpole	(p, v, ϕ, κ) $p \in [-2.4, 2.4]$ $v \in [-inf, inf]$ $\phi \in [-41.8, 41.8]$ $\kappa \in [-inf, inf]$	$< 0, 1 >$	DQN $(4 \times 24 \times 48 \times 2)$	10^{-3}	300
Acrobot	$(p, v, \phi, \kappa, \omega, \psi)$ $p, v, \phi, \kappa \in [-1, 1]$ $\omega \in [-12.567, 12.567]$ $\psi \in [-28.274, 28.274]$	$< 0, 1, 2 >$	DQN $(6 \times 24 \times 48 \times 3)$	10^{-3}	500

Strategically-Timed attack (ST attack) [18], the Q attack[1] [14] and the Multi-Objective RL attack (MO attack) [19]. Unlike the uniform attack, both the ST and Q attacks try to do as much damage to the victim as possible by reducing the number of attacks. The ST attack is triggered only when the discrepancy between the most and least favored action surpasses a predetermined threshold. Additionally, the Q attack computes the highest Q value for each state and launches an attack if this value surpasses a threshold. Hence, both the ST and Q attack necessitate a threshold, with thresholds set at 0.3 and 1.4, respectively. Finally, we compare these attacks to the more advanced MO attack, which aims to undermine the victim's policy in the long term. For this purpose, MO-attack aims to achieve two goals: maximizing the damage to the victim's policy and minimizing the cost of attacks in order to avoid detection. To configure the type of attack, MO-attack also necessitates defining a weight, denoted by w, for its two optimization metrics. If $w = 0$, the attack prioritizes cost optimization, whereas if $w = 1$, the adversary prioritizes optimizing the damage inflicted on the victim. We use two different versions of the MO, one that prioritizes causing maximum damage to the victim with a weight of 0.7, and another that aims to minimize the cost of the attack with a weight of 0.2. We would like to clarify that the chosen attack strategies are state of the art in terms of minimizing the number of attacks to be executed and preventing the attacker from launching continuous attacks, which would make them easier to detect. Furthermore, in the context of RL, few attack strategies have been proposed to date, and the chosen ones represent a broad spectrum of strategies that allows for a comprehensive analysis of the detection capabilities of the proposed method.

In these attack strategies, an attack is defined as the addition of noise, denoted as δ, to the original state. The range of noise values used in this paper is described in Table 2, and within this range, there are six different attacks defined. Some of them are classified as minor, while others are deemed more damaging perturbations. This analysis serves to determine if our approach can detect both small and large perturbations.

The rationale for the specific parameter values in Table 2 is rooted in their ability to ensure a correct balance between the disruption inflicted to the behavior policy of the victim and the associated attack cost: it is easy for our approach to identify high-level noise attacks that cause significant disturbance to the victim, while simultaneously acknowledging its limitations in detecting low-level noise attacks that, in fact, may not disturb the victim. The values presented in Table 2 have been chosen to establish a correct balance between these disparate scenarios. Naturally, the more aggressive the perturbation, the greater the cost. However, the only multi-objective strategy which determines the most effective attack to disrupt the victim by minimizing attack costs is the preferred approach. Other methods randomly select from predefined attacks, ignoring cost considerations.

6.2 Single observations vs. transitions

In this section, we demonstrate that detection mechanisms relying solely on single observations are incapable of detecting adversarial attacks' impact on the environment's dynamics. We note that single-observation-based detection methods are most popular for detecting adversarial attacks [11, 25–

[1] Kos and Song [14] use the Q attack only when the value function surpasses a certain threshold, thus only targeting crucial moments to disrupt the victim. We are build upon the same idea, but for practical purposes we employ the action-value function instead of the value function.

Table 2 Range of δ values, which represents the amount of noise added to the original state

Grid World	Mountain Car	Carpole	Acrobot
$\delta\ (x, y)$	(p, v)	(p, v, ϕ, κ)	$(p, v, \phi, \kappa, \omega, \psi)$
$x, y \in [-1, 1]$	$p, v \in [-0.025, 0.025]$	$p \in [-0.25, 0.45]$	$p, v, \phi, \kappa \in [-0.025, 0.025]$
		$v \in [-0.50, 0.26]$	$\omega \in [-0.31, 0.31]$
		$\phi \in [-0.43, 0.25]$	$\psi \in [-0.71, 0.71]$
		$\kappa \in [-0.05, 0.81]$	

27]. However, we show in this section that such methods are ineffective in an RL context.

Figure 3 displays an example 3x3 maze where an agent must navigate to the goal box marked in green, denoted by the letter G on the grid. The agent has acquired an optimal policy, π, and consequently, in a scenario lacking interference, it would reach its goal in four steps. In this example, we assume that an adversary begins injecting attacks at time step 160. This basic attack involves altering a single state: whenever the victim enters state s_{11}, the adversary deceives it into thinking it entered state s_{01} instead of s_{11}. As a result, the attack generates the adversarial transition $\langle s_{21}, up, s_{01}, r\rangle$ rather than the legitimate transition $\langle s_{21}, up, s_{11}, r\rangle$.

In this scenario, it is assumed that there are two detectors based on the approach outlined in Section 5. One focuses on transitions, while the other focuses solely on single states. For simplicity, both detectors have the same number of centroids in $\mathcal{C}$ as there are legitimate transitions or states in the task. Figure 1(b) displays $\Gamma^{\mathcal{C}}$ values for each detection mechanism. It indicates the distance of transition or state at the time step t from the closest centroid in $\mathcal{C}$. We observe that the state-based detector does not exhibit any bias in $\Gamma^{\mathcal{C}}$ due to this attack, making it undetectable (red line in Fig. 1(b)). If the adversarial transition $\langle s_{21}, up, s_{01}, r\rangle$ is analyzed statically, considering the states individually, both s_{21} and s_{11} belong to the task's state distribution, making the argument valid. However, the transition-based detection mechanism (blue line in Fig. 3(b)) highlights that the sequence $\langle s_{21}, up, s_{01}, r\rangle$ violates the environmental dynamics. Hence, in a sequential decision-making task, it is crucial to examine not only whether the states belong to the original distribution of states [11, 23], but also the consistency in the environment's dynamics.

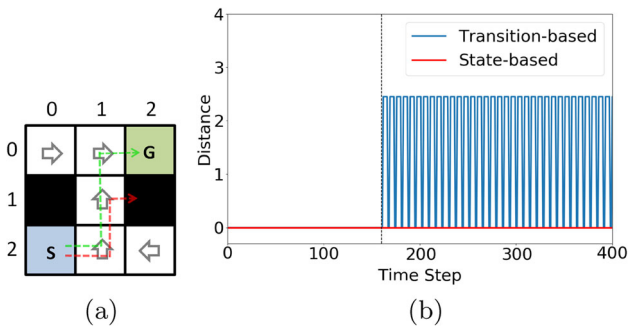


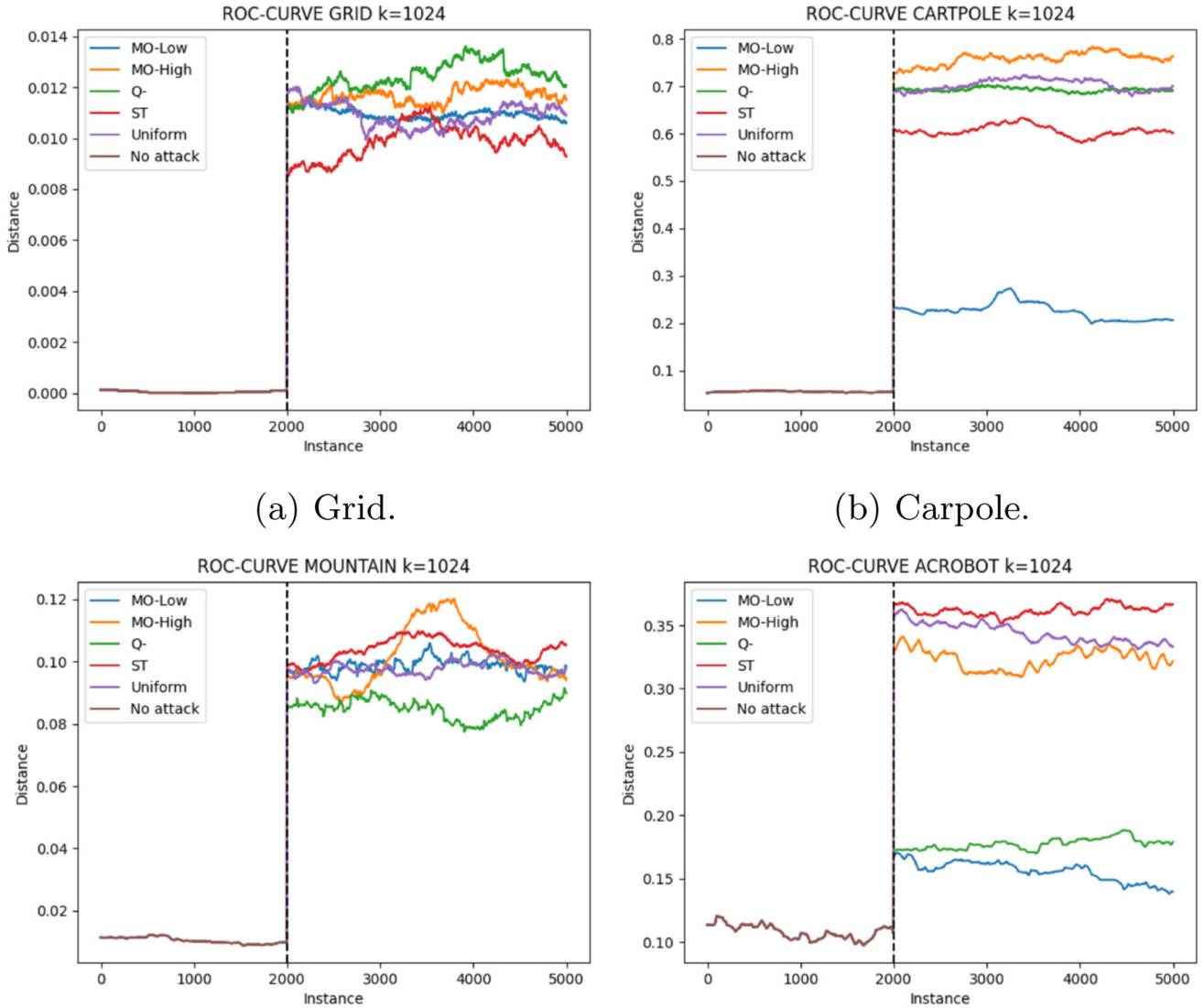
Fig. 3 (a) Deterministic 3 × 3 grid world, and (b) graphical representation of the sequences $\Gamma^{\mathcal{C}}$ of the state-based and transition-based detection mechanisms

Thus, defenses relying solely on recognizing known states would not identify this type of attack. However, our approach, which analyzes the entire transition would be able to detect that this transition has never occurred in a system free of attacks, and would therefore be able to detect that the agent is the victim of a falsification of the transitions it receives. In addition, analyzing the entire transition would enable us to identify all attacks on the RL systems depicted in the Fig. 1.

6.3 Context change detection: from *Free-of-attack* to *under-attack* context

The objective of this section is to verify the capability of the proposed method in detecting a change in context from a *free-of-attack* scenario to an *under-attack* scenario. We assume that a policy π has already been learned, and Algorithm 1 employs this policy to generate the transition space partition $\mathcal{C}$ and a list of thresholds β for each of the suggested domains. We assume that a policy π has already been learned, and Algorithm 1 employs this policy to generate the transition space partition $\mathcal{C}$ and a list of thresholds β for each of the suggested domains.

Figure 4 illustrates how the $\Gamma^{\mathcal{C}}$ sequence for each domain evolves over time. To enhance the clarity of the illustration, we calculate a simple moving average of 1000 transitions in order to smooth the trend in the four domains. Additionally, the vertical dashed line denotes the point where the adversary begins injecting attacks. This demarcation point signifies the transition from a context free of attacks to one that is under attack. From the vertical line in Fig. 4, a line for each attack type is plotted. A clear change in the trend of the $\Gamma^{\mathcal{C}}$ sequence due to the adversary's attacks is evident. It is observed that the MO attack causes the highest distortions in Fig. 4 as it produces transitions that are more dissimilar from the training set than the other attacks. The Uniform, ST, and Q attacks produce comparable results. These attacks employ the same strategy for perturbation injection, but the ST and Q attacks differ by performing fewer attacks, only when their metric exceeds a threshold. Our method can produce diverse signals



(a) Grid. (b) Carpole.

(c) Mountain car. (d) Acrobot.

Fig. 4 Evolution of the distances in the sequence $\Gamma^{\mathcal{C}}$ for each domain

when the agent is being attacked in any test environment. This is due to the fact that we analyze the sequential nature of the transitions rather than focusing strictly on the perceived states.

The objective is to identify any alteration in the sequence trend $\Gamma^{\mathcal{C}}$ as soon as possible. Several approaches may be used to accomplish change point detection, but we opted for the model fitting method presented in Section 5.2. When the logical expression $attack = [d_{\tau_t} > \beta_j]$ evaluates to true, transition τ_t is designated as an adversarial transition, and execution is stopped. Table 3 displays the accuracy of our detector with the given logical expression and $k = 1024$ clusters. The value of k adequately covers training transitions in the stochastic grid domain, resulting in the detection of almost all attacks.

The results in Table 3 reveal successful detection of the most harmful attacks generated by all analyzed strategies. Notably, the "mountain car" domain poses the most difficult challenges in detecting these disturbances, given its two variables and minor variations in transitions compared to the original distribution. In contrast, MO attack is detected in all domains with significant success. This strategy follows optimality criteria, leading to an attack policy of diverting the victim from its initial trajectory. The success of these attacks results from the victim's movement through less-visited regions of the state space, providing more diverse transitions

Table 3 Accuracy of attacks detected using the radius of each cluster as a threshold with 1024 clusters

	MO (high)	MO (low)	Uniform	Q-value	ST-attack
Grid	1.00	1.00	0.84	1.00	1.00
Carpole	0.85	0.86	0.80	0.77	0.69
Mountain car	0.74	0.80	0.61	0.69	0.63
Acrobot	0.88	0.86	0.83	0.91	1.00

during analysis. Additionally, this approach involves unnecessary attacks that have no impact on the initial transition causing inaccurate anomalies that our detector is unable to identify.

There are two methods to enhance success rates: modifying the threshold parameter, β, to differentiate the adversarial transitions more effectively, or increasing the number of clusters for these domains. Subsequently, we will aim to identify these attacks by separately modifying β in the next subsection.

6.4 Detection of adversarial transitions from a classification perspective

In this section, we will evaluate the quality of our detection mechanism. Specifically, we will measure the classifier's ability to distinguish between adversarial and non-adversarial transitions. We analyze accuracy by examining the threshold β and the number of clusters. In this evaluation, we assume that the parameter β_j remains constant for all clusters to simplify the process. To conduct a sensitivity analysis, we calculate the ROC curve, which displays the true positive rate (TPR) along the y-axis and the false positive rate (FPR) on the x-axis. This approach visualizes the detection system's performance as the threshold β increases. Therefore, each point on the ROC curve represents a different threshold at which our detector produces different results in terms of true positive rate versus false positive rate. Initially, when β is at 0, all transitions are identified as adversarial. As the value of β increases, a higher number of transitions are classified as non-adversarial, leading to a reduction in false positives by correctly recognizing these transitions as non-adversarial. Nevertheless, some attacks might be incorrectly identified. Therefore, we utilize ROC curves to determine the optimal value of β. Eventually, β reaches its maximum value and the detection system classifies all transitions as non-adversarial. We perform an exhaustive search for all thresholds, and if the area under the curve (AUC) is equal to 1, it indicates that the detection system is effectively classifying all transitions. Conversely, an AUC close to 0.5 indicates that the classifier is behaving randomly.

Figure 5 contains all the results of our defense. Each row in Fig. 5 contains the results of each domain, i.e., we show the results of grid, carpole, mountain car and acrobot respectively. Additionally, each column shows the results grouped by each clustering configuration, we show the results for $k = 64, 256, 1024$ respectively. In each figure we plot a line for each attack evaluated: MO attack with $w = 0.7$ (blue), MO attack with $w = 0.2$ (orange), Uniform (purple), $Q-$ (green) and ST attacks (red). In addition to the ROC curve, we attach the AUC score of each line in the legend. This allows us to distinguish the configuration with the best performance for each attack and domain.

As the number of clusters k increases, we obtain better results. Nonetheless, the value of k increases the complexity of building the transition partition. For this reason, we suggest that a higher value of k is not necessary. If we focus on the last column in Fig. 5 ($k = 1024$), in general, our defense obtains promising results in all the domains detecting all kinds of attacks. We achieve AUC scores over 0.8 in the grid and the carpole domains. Nevertheless, smooth variations in the transitions of the mountain car are enough to mislead the victim. From the point of view of our detector, it is more difficult to distinguish the instances perturbed by smooth attacks. As we have analyzed the signal generated for all domains, when introducing perturbations in both *free-of-attack* and *under-attack* scenarios, in the domain of acrobot, it is observed that the signal generated in a free-adversary environment is higher than in other domains. However, upon introducing attacks, there is a noticeable increase in this signal. This confirms that, as with other domains, the perturbed states differ from the original ones and thus our detection system is capable of detecting these perturbations. For this reason, we obtain worse AUC scores in this domain especially, under the Uniform and the ST attacks, as we observe in Fig. 5(i) and (l).

Obviously, as the number of clusters increases, the original transitions are better distinguished from the adversarial ones. We show that this occurs in the grid domain, comparing Fig. 5(c) where the number of clusters is almost similar to the number of undisturbed transitions. In other words, a low number of centroids means that some regular instances get a farther distance from their nearest centroid, as we observe in Fig. 5(a). Then, the detector may classify these instances as attacks, i.e., as adversarial transitions, when they are non-adversarial ones, increasing the number of false positives. Therefore, in these cases, the AUC score is lower. In general, MO attacks are easier to detect. Especially the version which maximizes the harm to the victim. This attack learns the best

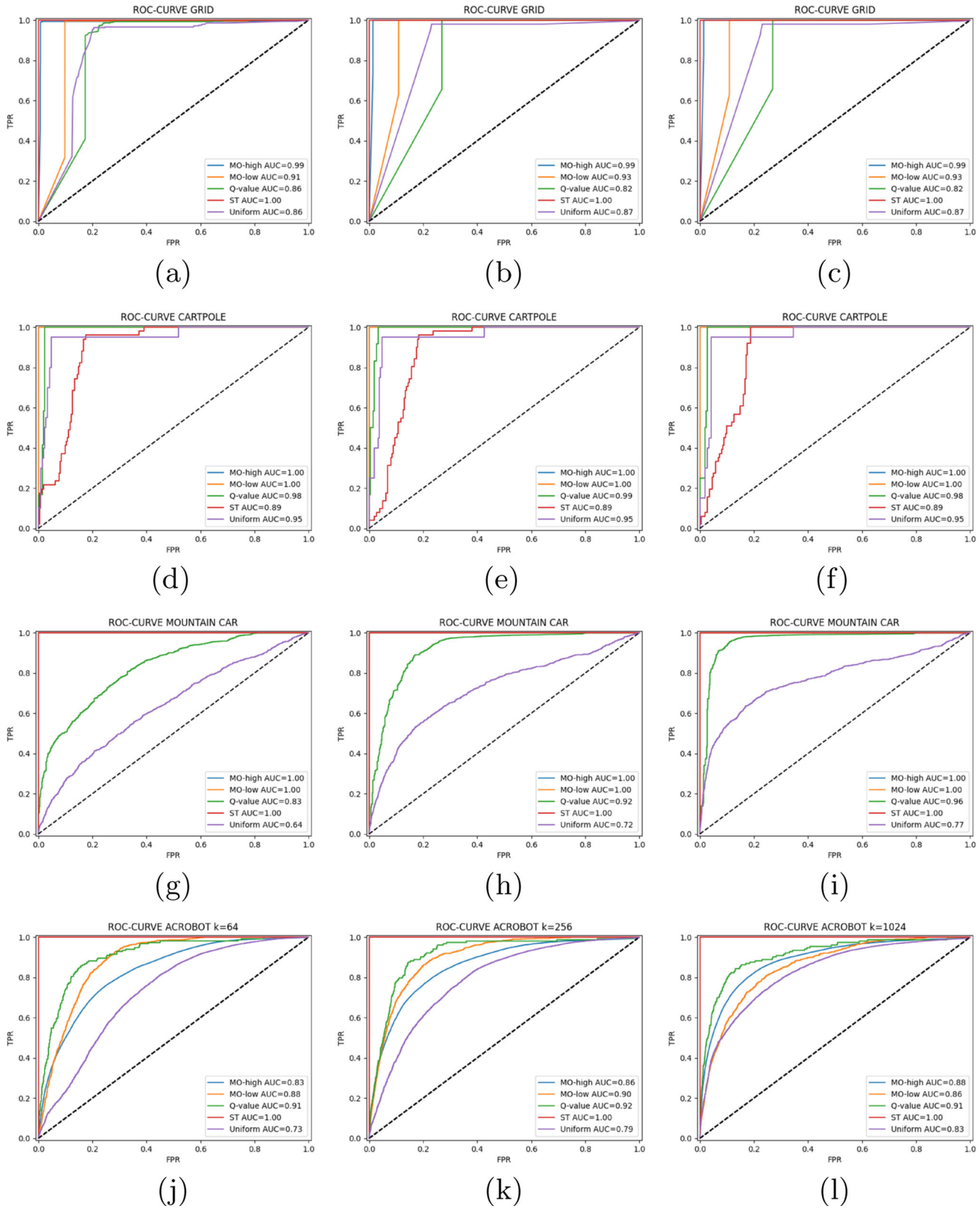
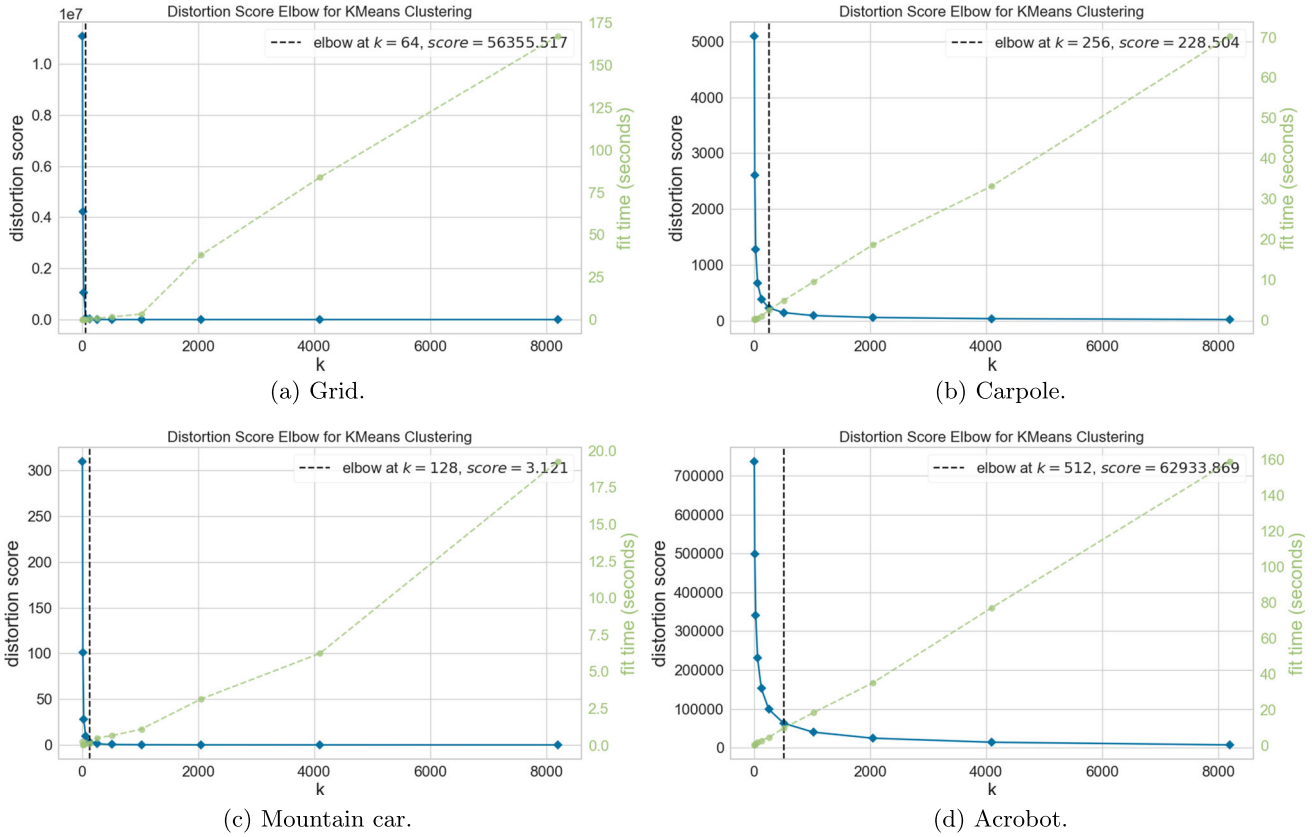


Fig. 5 ROC-curve experimentation results. Each plot contains the five attacks evaluated: MO attack with $w = 0.7$ (blue), MO attack with $w = 0.2$ (orange), Uniform (purple), $Q-$ (green) and ST attacks (red). The Area Under the Curve (AUC) of each ROC curve is in the legend. Each column shows the results using 64, 256 and 1024 clusters, respectively

perturbation to drive the victim to undesired states. Once the victim is in an unknown place, the enemy stops attacking. However, the victim generates transitions that differ from the training dataset because these transitions do not belong to the instances used to learn. In this way, our distance metric generates larger values that exceed the threshold and, it produces that our detector classifies these transitions as attacks increasing the ratio of false positives. Such false positives can be reduced by increasing the exploration rate during training to capture more trustworthy examples. After the MO attack, $Q-$ attack is the next attack more detectable. We obtain AUC scores over 0.8 in all the domains. This attack attempts to mislead the victim in states close to its goal. In contrast to MO, $Q-$ attack does not drive the victim to undesired locations. This attack deviates the victim at the end of the episode, allowing our detector to identify these anomalies correctly. ST attack chooses an attack randomly when the difference between the best and the least preferred action exceeds a threshold. As equal to the MO-attack, the enemy attacks in some critical points, leading the victim to uncharted states. As a result, we obtain lower AUC scores in this attack. The same occurs in the Uniform attack. The AUC score of the Uniform attack is over 0.8 in the grid and carpole domain. However, the performance of our detector decreases in the Mountain Car. The reason is that some of these perturbations do not alter enough the trajectory of the victim. This type of perturbation makes it more difficult to detect the Uniform strategy than the rest of the evaluated attacks.

In the Acrobot domain, we achieve strong results, with a success rate exceeding 80%, in identifying attacks generated against the attack strategies tested. Although the perturbations introduced in this domain are small, they are applied to a larger number of variables compared to the other domains. For this reason, the anomaly detection results obtained in this domain are very high. Consequently, as the threshold β increases, both the true positive ratio and the false positive ratio experience parallel increments. Furthermore, even our approach is successful against the multi-objective attack strategy. Compared to the Q-value and ST strategies, the other attacks create more disruptions during the first steps of the episode. There are numerous additional transitions in the initial steps of episodes with distinct domain initialization. Consequently, our detection system can create a significant number of clusters in the beginning part of the domain and less toward the end. For this reason, attack strategies that initiate a greater number of attacks at the start are more chal-



(a) Grid. (b) Carpole.

(c) Mountain car. (d) Acrobot.

Fig. 6 Elbow method for our domains: (a) Grid, (b) Cartpole, (c) Mountain car, and (d) Acrobot

lenging to detect. This is due to the fact that the original transitions bear a closer resemblance to the anomalous ones.

6.5 Ablation study of *k*

Selecting the appropriate value for k in the k-means clustering algorithm is a crucial step that significantly influences the outcome of the clustering process. This choice essentially determines how the data is grouped and organized into distinct clusters. If k is incorrectly chosen, it can lead to inaccurate representation of the data's underlying structure. For instance, a very high k might result in an excessive number of clusters, making it difficult to extract meaningful insights. On the other hand, choosing a very low k could oversimplify the representation, overlooking valuable patterns and groupings in the data. Striking the right balance with k is essential to derive meaningful and actionable insights from the clustering process, which is vital for our purpose.

In our approach, dealing with the under fitting problem holds heightened importance. Inaccurate representation of the data structure could lead us to misclassify typical transitions as anomalies, significantly inflating the count of false positives. Consequently, if a notable surge in false positives is noted, adjusting the k value to generate additional prototypes is a viable approach. The objective is to enhance the fidelity of representing the initial data and achieve a more precise differentiation between normal transitions and anomalies.

To determine the optimal k for our detector, we execute an elbow method to determine the optimal value of centroids to use in a k-means clustering algorithm. This method find the equilibrium between the number of clusters needed to reduce the distortion of the clustering points and the time to execute the algorithm. The optimal k is typically selected at this point, balancing the trade-off between model complexity and clustering quality. The elbow method provides a visual aid and a quantitative basis for selecting a suitable number of clusters, enhancing the effectiveness and interpretability of clustering results in various applications. Plots in Fig. 6 illustrates that for our four domains, the number of optimal clusters is always less than 1024, following the elbow method. For this reason, we opted for 1024 clusters due to the sufficiency of the distortion obtained in identifying adversarial transitions more effectively in all domains covered in this paper, and, in addition, computing the cluster set does not entail an excessively high time investment.

In addition to these plots, we also generate Table 4 with the average of the distortion and its standard deviation metrics for these set of clusters. The table presents the results of 10 executions with randomly initialized centroids. Table 4 demonstrates that the distortion is significantly reduced when using 1024 clusters, regardless of random initialization. In all cases, the standard deviation approaches zero, thereby guaranteeing robust detection of adversarial transitions, as demonstrated in Sections 6.3 and 6.4.

The graphs and tables showcased in this section provide clear evidence that, within the chosen domains, distortion noticeably diminishes once the number of clusters reaches 1024. Hence, we have opted to employ this specific k-value for our approach. Given that our proposed approach is designed to be applicable across various domains, it becomes imperative to undertake a comparable analysis in order to ascertain the suitable k-value for preventing under fitting in the data structure of each distinct domain.

7 Conclusions

This paper describes a novel clustering-based approach for detecting outlier transitions in RL. We evaluate our detector against four state-of-art attacks in three well-known domains. Next, we summarize the main conclusions found in this paper:

(i) A novel framework for attack detection. The main contribution of this paper is a novel framework for attack detection based on the perturbations these attacks produce in the transition and reward dynamics that the victim per-

Table 4 Distortion generated for different k values in the proposed domains

Domains				
k-value	Grid	Carpole	Mountain car	Acrobot
8	1.606.202,8 ± 31.307,1	7.582,1 ± 218,8	338,9 ± 15,4	99.099,4 ± 274,4
16	306.451,4 ± 24.823,7	3.186,0 ± 107,0	111,6 ± 3,7	66.985,2 ± 280,4
32	67.827,8 ± 3.674,1	1.292,7 ± 14,4	30,5 ± 0,3	42.092,5 ± 318,6
64	20.793,1 ± 572,7	613,8 ± 8,1	9,8 ± 0,2	25.867,1 ±185,1
128	3.740,5 ± 82,8	291,3 ± 2,0	3,4 ± 0,1	15.856,7 ± 65,1
256	444,6 ± 7,9	139,6 ± 0,5	1,3 ± 0,1	9.768,1 ± 39,9
512	0,0 ± 0,0	76,5 ± 0,2	0,5 ± 0,0	5.648,1 ± 9,8
1.024	0,0 ± 0,0	40,9 ± 0,1	0,2 ± 0,0	2995,3 ± 9,0
2.048	0,0 ± 0,0	22,4 ± 0,1	0,1 ± 0,0	1223,5 ± 6,6

ceives from the environment. The experiments in Section 6.3 demonstrate that the victim goes from a *free-of-attack* context to an *under-attack* context whenever an adversary begins to inject attacks, and it is precisely this context change that must be detected to prevent catastrophic consequences. Therefore, this novel perspective opens the door to the application of change-point detection approaches that are able to identify changes in the transition and reward dynamics perceived for the victim.

(ii) A novel model-free cluster-based detector. Clustering of the transition space allows, on the one hand, not having to know the transition and reward dynamics of the environment, and, on the other hand, to transform a multi-variate detection problem into a more compact and tractable uni-variate one as described in Section 5. This is a significant advantage concerning other change-point detection approaches [6, 7, 28].

(iii) Exploitation of the coherence of the environmental dynamics. In contrast to the majority of the previous works which, focus on detecting single *adversarial observations*, we analyze *adversarial transitions*. The evaluation in Section 6.2 demonstrates that detectors focused on transitions instead of single observations can capture the dynamics of a sequential decision task. Therefore, transition-based detectors exploit the coherence of the transition and reward functions that make them better for adversarial attack detection in the context of RL.

(iv) Sensitivity of the proposed approach to the parameter β. Obviously, the success of the proposed approach is subject to the radius β defined for each of the clusters. It is mandatory to detect attacks as soon as possible in safety-critical domains. So this paper suggests that this parameter should be heuristically predefined before system deployment. In this case, β is tuned as described in (6), but other initialization could be investigated. Notwithstanding the above, we have also analyzed the sensitivity of the detection approach to β by using ROC curves (Section 6.4). ROC curves make a sweep of the β values and return for each one the rate of true and false positives. Therefore, we can easily choose the best threshold after plotting the ROC curve. Such *a posteriori* analysis could be interesting for non-safety-critical domains.

(v) A complex attack is more difficult to detect. As we show in the evaluation section, if the attack drives the victim to undesired or unexplored states, the victim will generate a higher amount of non-adversarial transitions, which our detector classifies as adversarial ones. Therefore, the number of false positives increases, and the performance of our detector decreases.

As future work, we would extend our approach to analyze the performance of other change-point detection techniques. We also would implement some methods to reconstruct the attacked transitions to allow the victim to continue its trajectory through its goal.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This research is granted by Repsol, the Spanish Government (Ministerio de Economia y Empresa) and FEDER, and EU. EU funds under project PID2021-127647NB-C21 and PDC2022-133597-C43 from MCIN/AEI/10.13039/501100011033, by the ERDF "A way of making Europe" and Next Generation EU/ PRTR and by the Madrid Government under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17) in the context of the V PRICIT (Regional Programme of Research and Technological Innovation). This research was also supported in part by AEI Grant PID2020-119367RB-I00.
This research was funded in partially by JPMorgan Chase & Co. Any views or opinions expressed herein are solely those of the authors listed, and may differ from the views and opinions expressed by JPMorgan Chase & Co. or its affiliates. This material is not a product of the Research Department of J.P. Morgan Securities LLC. This material should not be construed as an individual recommendation for any particular client and is not intended as a recommendation of particular securities, financial instruments or strategies for a particular client. This material does not constitute a solicitation or offer in any jurisdiction. Funding for APC: Universidad Carlos III de Madrid (Agreement CRUE-Madroño 2024)

Data Availability The datasets generated during and/or analyzed during the current study are available in the Adversarial detector repository, https://github.com/rmajadas/Adversarial-detector

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Chen T, Liu J, Xiang Y, Niu W, Tong E, Han Z (2019) Adversarial attack and defense in reinforcement learning-from AI security view. Cybersecur. 2(1):11. https://doi.org/10.1186/s42400-019-0027-x
2. Behzadan V, Munir A (2017) Vulnerability of deep reinforcement learning to policy induction attacks. In: Proceedings of the international conference on machine learning and data mining in pattern recognition. Lecture Notes in Computer Science, vol 10358, pp 262–275. Springer, New York, NY, USA. https://doi.org/10.1007/978-3-319-62416-7_19
3. Sharif M, Bhagavatula S, Bauer L, Reiter MK (2016) Accessorize to a crime: real and stealthy attacks on state-of-the-art face recog-

nition. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, Vienna, Austria, pp 1528–1540. https://doi.org/10.1145/2976749.2978392. Accessed 24–28 Oct 2016
4. Kurakin A, Goodfellow IJ, Bengio S (2017) Adversarial examples in the physical world. In: 5th International conference on learning representations, ICLR 2017, April 24-26, Workshop Track Proceedings. OpenReview.net, Toulon, France. https://openreview.net/forum?id=HJGU3Rodl
5. Deng Y, Zheng X, Zhang T, Chen C, Lou G, Kim M (2020) An analysis of adversarial attacks and defenses on autonomous driving models. In: 2020 IEEE International conference on pervasive computing and communications (PerCom), pp 1–10. https://doi.org/10.1109/PerCom45495.2020.9127389
6. da Silva BC, Basso EW, Bazzan ALC, Engel PM (2006) Dealing with non-stationary environments using context detection. In: Proceedings of the 23rd international conference on machine learning (ICML), pp 217–224. Association for Computing Machinery, New York, USA. https://doi.org/10.1145/1143844.1143872
7. Truong C, Oudre L, Vayatis N (2020) Selective review of offline change point detection methods. Signal Process 167. https://doi.org/10.1016/J.SIGPRO.2019.107299
8. Ghosh BK, Sen PK (1991) Handbook of Sequential Analysis
9. Basseville M, Nikiforov I (1993) Detection of Abrupt Change: Theory and Application vol 15
10. Zamba K, Hawkins DM (2006) A multivariate change-point model for statistical process control. Technometrics 48(4):539–549
11. Grosse K, Manoharan P, Papernot N, Backes M, McDaniel P (2017) On the (statistical) detection of adversarial examples. arXiv:1702.06280
12. Canonaco G, Restelli M, Roveri M (2020) Model-free non-stationarity detection and adaptation in reinforcement learning. In: European conference on artificial intelligence (ECAI), pp 1047–1054 IOS Press
13. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. IEEE Trans Neural Networks 9(5):1054–1054. https://doi.org/10.1109/TNN.1998.712192
14. Kos J, Song D (2017) Delving into adversarial attacks on deep policies. In: 5th International Conference on Learning Representations, ICLR 2017, April 24-26, Workshop Track Proceedings. OpenReview.net, Toulon, France. https://openreview.net/forum?id=BJcib5mFe
15. Pattanaik A, Tang Z, Liu S, Bommannan G, Chowdhary G (2018) Robust deep reinforcement learning with adversarial attacks. In: Proceedings of the 17th international conference on autonomous agents and multiAgent systems, pp 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems
16. Huang SH, Papernot N, Goodfellow IJ, Duan Y, Abbeel P (2017) Adversarial attacks on neural network policies. In: 5th International conference on learning representations, ICLR 2017, April 24-26, Workshop Track Proceedings. OpenReview.net, Toulon, France. https://openreview.net/forum?id=ryvlRyBKl
17. Pinto L, Davidson J, Sukthankar R, Gupta A (2017) Robust adversarial reinforcement learning. In: Precup D, Teh YW (eds.). Proceedings of the 34th international conference on machine learning, ICML 2017, 6-11 August 2017. Proceedings of Machine Learning Research, vol 70, pp. 2817–2826. PMLR, Sydney, NSW, Australia. http://proceedings.mlr.press/v70/pinto17a.html
18. Lin Y-C, Hong Z-W, Liao Y-H, Shih M-L, Liu M-Y, Sun M (2017) Tactics of adversarial attack on deep reinforcement learning agents. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI-17, pp 3756–3762. https://doi.org/10.24963/ijcai.2017/525
19. García J, Majadas R, Fernández F (2020) Learning adversarial attack policies through multi-objective reinforcement learning. Eng Appl Artif Intell 96:104021. https://doi.org/10.1016/j.engappai.2020.104021
20. Roy A, Xu H, Pokutta S (2017) Reinforcement learning under model mismatch. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett, R. (eds.). Advances in neural information processing systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 3043–3052. https://proceedings.neurips.cc/paper/2017/hash/84c6494d30851c63a55cdb8cb047fadd-Abstract.html
21. Everitt T, Krakovna V, Orseau L, Hutter M, Legg S (2017) Reinforcement learning with a corrupted reward channel. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence (IJCAI), pp 4705–4713. https://doi.org/10.24963/ijcai.2017/656
22. Goldblum M, Fowl L, Feizi S, Goldstein T (2020) Adversarially robust distillation. In: Proceedings of the AAAI conference on artificial intelligence, 2020, New York, USA, pp 3996–4003. https://aaai.org/ojs/index.php/AAAI/article/view/5816. Accessed 7–12 Feb 2020
23. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: Bengio Y, LeCun Y (eds.). 3rd International conference on learning representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings. arXiv:1412.6572. Accessed 7–9 May 2015
24. Papernot N, McDaniel PD, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. In: IEEE Symposium on security and privacy, SP 2016, pp 582–597. IEEE Comput Soc, San Jose, CA, USA. https://doi.org/10.1109/SP.2016.41. Accessed 22–26 May 2016
25. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
26. Gong Z, Wang W (2023) Adversarial and clean data are not twins. In: Bordawekar R, Shmueli O, Amsterdamer Y, Firmani D, Kipf A (eds.) Proceedings of the sixth international workshop on exploiting artificial intelligence techniques for data management, aiDM@SIGMOD 2023, pp 6–165. ACM, Seattle, WA, USA. https://doi.org/10.1145/3593078.3593935. Accessed 18 June 2023
27. Lee K, Lee K, Lee H, Shin J (2018) A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Advances in neural information processing systems, pp 7167–7177
28. Alegre LN, Bazzan ALC, da Silva BC (2021) Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. In: Proceedings of the 20th international conference on autonomous agents and multi-Agent systems. AAMAS '21, pp 97–105
29. Komorniczak J, Zyblewski P, Ksieniewicz P (2022) Statistical drift detection ensemble for batch processing of data streams. Knowl Based Syst 252:109380. https://doi.org/10.1016/J.KNOSYS.2022.109380
30. Jain M, Kaur G, Saxena V (2022) A k-means clustering and SVM based hybrid concept drift detection technique for network anomaly detection. Expert Syst Appl 193:116510. https://doi.org/10.1016/J.ESWA.2022.116510
31. Hushchyn M, Ustyuzhanin A (2021) Generalization of change-point detection in time series data based on direct density ratio estimation. J Comput Sci 53:101385. https://doi.org/10.1016/J.JOCS.2021.101385

Rubén Majadas graduated from the University Carlos III of Madrid. He is currently a PhD student, focusing on adversarial reinforcement learning methods from both attack and defense perspectives for his doctoral dissertation. His research interests encompass machine learning, particularly reinforcement learning, optimization, and adversarial networks.

Javier García is an associate professor of the Electronics and Computer Science Department at the Universidad de Santiago de Compostela, since september 2023. He obtained a PhD that received the distinguished Thesis Award at the Computer Science Department of Universidad Carlos III de Madrid in 2013. He was also a postdoctoral researcher under the Talent Attraction Program from the Comunidad de Madrid since 2016 until 2020 in the Computer Science Department of the Universidad Carlos III de Madrid. He has more than 40 journal and conference papers, mainly in the field of machine learning, automated planning and robotics. Currently, he is associate editor of the journal IEEE Robotics and Automation Letters and member of the editorial board of the journal Machine Learning.

Fernando Fernández is an associate professor of the Computer Science Department at Universidad Carlos III de Madrid, since october 2005. He received his Ph.D. degree in Computer Science from University Carlos III of Madrid (UC3M) in 2003. In the fall of 2000, Fernando was a visiting student at the Center for Engineering Science Advanced Research at Oak Ridge National Laboratory (Tennessee). He was also a postdoctoral fellow at the Computer Science Department of Carnegie Mellon University since october 2004 until december 2005 and Visiting Researcher at Texas Robotis, University of Texas at Austin in 2022-2023. He is the recipient of a pre-doctoral FPU fellowship award from Spanish Ministry of Education (MEC), a Doctoral Prize from UC3M, and a MEC-Fulbright postdoctoral Fellowship as well as the 2020 and 2021 JPMorgan AI Research Awards. He has more than 80 journal and conference papers, mainly in the field of machine learning, automated planning and robotics. He is interested in intelligent systems that operate in continuous and stochastic domains. He is the Director of the Planning and Learning Group of the Computer Science Department at UC3M. He is also co-founder and CSO of Inrobocs Social Robotics, where social assistive robots are deployed in rehabilitation hospitals.