Spiking Pitch Black: Poisoning an Unknown Environment to Attack Unknown Reinforcement Learners

Hang Xu
Nanyang Technological University
Singapore
hang017@e.ntu.edu.sg

Xinghua Qu
ByteDance AI Lab
Singapore
quxinghua17@gmail.com

Zinovi Rabinovich
Nanyang Technological University
Singapore
zinovi@ntu.edu.sg

ABSTRACT

As reinforcement learning (RL) systems are deployed in various safety-critical applications, it is imperative to understand how vulnerable they are to adversarial attacks. Of these, an environment-poisoning attack (EPA) is considered particularly insidious, since environment hyper-parameters are significant factors in determining an RL policy, yet prone to be accessed by third parties. The success of EPAs relies on comprehensive prior knowledge of the attacked RL system, including RL agent's learning mechanism and/or its environment model. Unfortunately, such an assumption of prior knowledge creates an unrealistic attack, one that poses limited threat to real-world RL systems.

In this paper, we propose a Double-Black-Box EPA framework, only assuming the attacker's ability to alter environment hyper-parameters. Considering that environment alteration comes at a cost, we seek minimal poisoning in an unknown environment and aim to force a black-box RL agent to learn an attacker-designed policy. To this end, we incorporate an inference module in our framework to capture the internal information of an unknown RL system and, accordingly, learn an adaptive strategy based on an approximation of our attack objective. We empirically show the threat posed by our attack to both tabular-RL and deep-RL algorithms, in both discrete and continuous environments.

KEYWORDS

Reinforcement Learning; Security; Environment Poisoning

ACM Reference Format:
Hang Xu, Xinghua Qu, and Zinovi Rabinovich. 2022. Spiking Pitch Black: Poisoning an Unknown Environment to Attack Unknown Reinforcement Learners. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

The security of Reinforcement Learning (RL) has become increasingly significant due to the widespread deployment of RL systems in safety-critical applications, such as autonomous cars [13, 21, 27], smart energy systems [12, 14, 29] and healthcare systems [5, 6, 26]. However, RL policies are typically sensitive to training hyper-parameters [8, 15, 20], where a slight variation of these parameters may cause obvious performance difference. As a result, RL policies are vulnerable to being perturbed by poisoned training hyper-parameters. Among these parameters, environment hyper-parameters are most susceptible as they can be easily accessed by third parties. Therefore, to facilitate the formulation of secure strategies, a study of the threats posed by environment hyper-parameters is necessary.

Environment hyper-parameters, particularly in physical systems, are also termed *causal factors* or *hidden parameters*, such as gravity and friction of a surface [19, 22, 30]. These hyper-parameters can not be observed directly, but their effect can be 'felt' by the RL agent when interacting with the environment. This means that environment hyper-parameters affect how the environment responds to the RL agent's actions. Namely, they can be used to parameterize the environment transition dynamics [11, 19, 22] so that altering hyper-parameters leads to changes of the environment dynamics. In previous studies [1, 24, 34], such adversarial changes of environment dynamics, termed Environment-Poisoning Attacks (EPAs), assumed the internal information of an RL system to be known in advance, including: a) how the RL agent learns its policy, i.e., the learning algorithm and the policy model; b) how the environment responds to the agent's action, i.e., the environment dynamics; c) how hyper-parameters determine the environment dynamics, i.e., the environment causal mechanism. However, such information is generally private or unknown in most real-world RL-based applications, and we argue that these assumptions make an attack approach unrealistic. To alleviate such a limitation, we study a novel environment-poisoning attack that requires minimal attacker's prior knowledge of an RL system.

Specifically, we propose a Double-Black-Box Environment Poisoning Attack (DBB-EPA) approach, which achieves policy compulsion on an *unknown* RL agent in an *unknown* environment. To the best of our knowledge, this is the first environment-poisoning attack that does not rely on prior knowledge of both the agent's learning mechanism (i.e., policy training algorithm and policy model structure/parameters) and its environment model (i.e., transition and reward functions). Assuming only the ability to alter the environment hyper-parameters, our attack aims to force a black-box RL agent to learn an attacker-desired policy, with minimal and adaptive environment poisoning. To this end, we first investigate how to infer the internal information of an RL system, and then learn an adaptive attack strategy based on the approximation of our attack objective. Specifically, given observations of an RL agent's trajectories during its learning process, we jointly train: a) an Encoder-Dual-Decoder network that learns a low-dimensional latent representation of the RL system's internal information; b) an attack strategy, conditioned on the latent representation and environment hyper-parameters, that manipulates the RL agent's policy using minimal environment poisoning. In contrast to the existing EPAs [1, 24, 34] that are only effective in discrete state domains, our work provides a tractable approach to attack deep-RL (DRL) agents in continuous environments.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

In summary, the contributions of this paper are as follows:

- We propose an environment-poisoning attack approach in the double-black-box setting, where both the RL agent's learning mechanism and its environment model are unknown to the attacker – DBB-EPA.
- We design an Encoder-Dual-Decoder network to infer internal information of a black-box RL system and construct an adaptive attack strategy conditioned on the resultant latent representation by approximating our attack objective.
- We show that DBB-EPA achieves performance comparable to the white-box attack [34] on a navigation task in the grid world. We further evaluate our attack against a DRL agent on a control task in continuous domains, showing the feasibility and the scalability of DBB-EPA on more complex RL systems.

2 RELATED WORK

In this section, we provide an overview of training-time attacks against RL, followed by a specific literature review of recent advances in Environment-Poisoning Attacks (EPAs).

The objective of training-time attacks is to force an RL agent to learn a target policy designed by an attacker. To achieve this objective, the attacker poisons the agent's policy learning by perturbing feedback (e.g., reward signals and state observations) from the training environment. Accordingly, there are mainly two categories in training-time attacks: reward-poisoning attacks and environment-poisoning attacks (see e.g., [3, 10] for a review). For example, the reward poisoning is studied on both the off-line batch RL [17] and the online Q-learning algorithm [9, 36]. At the same time, Rakhsha et al. [24] study an environment-poisoning attack in cyclic RL tasks by directly manipulating the transition dynamics.

Notably, though, most training-time attacks are developed in the white-box setting, where the attacker has significant access to comprehensive knowledge of the RL system. In particular, the RL agent's environment model (i.e., transition dynamics and reward functions) and its learning mechanism (i.e., policy training algorithm and policy model structure/parameters) are assumed to be known by the attacker. Unfortunately, assuming such an omniscient attacker makes most attack approaches somewhat unrealistic, resulting in limited threats to RL systems in the real world.

Recently, however, researchers made a breakthrough in the development of realistic reward-poisoning attacks that can manipulate the RL policy *without* prior knowledge of the RL system [25, 31]. In particular, Rakhsha et al. [25] theoretically study a black-box reward-poisoning attack against no-regret RL algorithms. In turn, Sun et al. [31] propose a practical reward-poisoning algorithm for policy-based deep RL methods without knowledge of the environment. In this paper, we pursue the same motivation as [25, 31], but focus on realistic environment-poisoning attacks. More importantly, instead of directly altering environment dynamics, we investigate the threat posed by maliciously controlled environment hyper-parameters.

In this context, of particular interest is the Transferable Environment Poisoning Attack (TEPA) proposed in [34], which adaptively poisons environment hyper-parameters for a given RL algorithm – the white-box proxy. Xu et al. [34] demonstrate that TEPA is transferable and it successfully poisons RL agents that differ algorithmically from the white-box proxy. In fact, TEPA is thus applicable to black-box RL agents. However, the full knowledge of the training environment lies at the heart of TEPA as well. Thus, since environment-dynamics information is generally private or unknown in most real-world applications, TEPA is confined to white-box environments and, therefore, poses limited threat to complex RL systems. In this paper, grafting the advantages and framework of TEPA , we develop a novel attack approach that is independent from the prior knowledge of training-environment dynamics model and causal mechanism. To the best of our knowledge, this paper is the first to study EPAs without assumptions of prior knowledge of both the RL agent's training environment and its learning mechanism.

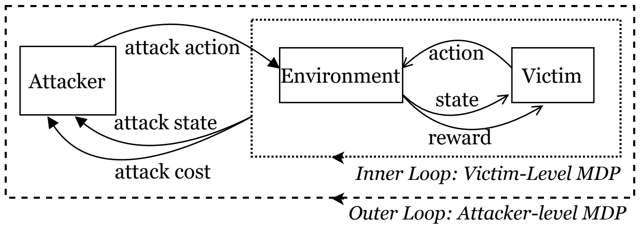


Figure 1: Attack framework

Now, it must be mentioned that there are RL approaches capable of identifying environment changes induced by hyper-parameter shifts, and constructing a robust behaviour strategy in the context of such changes [19, 22, 30]. The goal of these methods is to generate a behaviour useful across tasks and environments. However, they mostly disregard the possibility of a *constructive, strategic* adversary that modulates environment hyper-parameters. Thus, while we view robust algorithms as a key component in building white-box proxies in our future research, in the current paper we make the standard relaxation assumption of training-time RL attacks. Namely, we assume that the attacked RL agent is oblivious to the attack and continues to operate normally throughout the sequence of environment modifications.

3 PROBLEM STATEMENT

This section describes the attack framework proposed for environment poisoning. We first provide notations and preliminaries of the attack framework, followed by the mathematical description of our attack problem.

3.1 Notations and Preliminaries

We adopt a bi-level Markov Decision Process (MDP) architecture [34, 36], which is illustrated in Figure 1. The task of poisoning an RL agent's (i.e., victim) policy is performed by an another RL agent (i.e., the attacker) that operates on a different timescale from the victim. Specifically, with a particular attack frequency, the attacker manipulates the victim's training-environment hyper-parameters in response to the victim's learning progress. The attacker's objective is to achieve an optimal attack strategy that succeeds in poisoning the victim's policy while minimizing changes to the victim's environment. The attack framework is formally described as follows.

Victim-level MDP. The Markovian environment of a victim is denoted by the tuple $< S, A, T_e, r, d_0, \hat{\gamma} >$. Here, $s \in S$ is an environment state and $a \in A$ is the victim's action. $T_e(s'|s, a)$ represents the victim's dynamics function that tells the victim's state transition probabilities from s to s', given the action a and the environment hyper-parameter e. $r : S \times A \times S \rightarrow \mathbb{R}$ represents the victim's reward function. $d_0(s)$ is a distribution over the victim's initial states and $\hat{\gamma}$ is a discount factor. The victim aims to learn an optimal policy $\pi(a|s)$ that maximizes the cumulative discounted rewards $\sum_t^\infty \hat{\gamma}^t r_t$. It's trajectory τ is a sequence of state-action pairs generated by π. Additionally, the attacker-desired policy is represented as π^* and the corresponding desired trajectory is denoted as τ^*.

Attacker-level MDP. An attacker, which is regarded as an outer-loop RL agent as shown in Figure 1, treats the victim-level system as its dynamics environment. We describe the attacker's Markovian process by the tuple $< X, U, F, c, \gamma >$ as follows.

- X is the attacker's state space. $x_i \in X$ represents an attack state that contains information about the victim's behaviour policy and its environment dynamics at the i^{th} attack epoch.
- U is the attacker's action space. An attack action $u \in U$ represents a manipulation on the victim's environment hyper-parameter e. Here, e influences how the training environment responds to the victim's action, i.e., environment dynamics. e_i represents the poisoned environment which has been changed by aggregate attack actions $\{u_1, u_2, ..., u_i\}$. e_0 denotes the victim's natural environment.
- $F : X \times U \times X \rightarrow [0, 1]$ is the attacker's probabilistic state transition function. It captures how the victim updates its policy under effect of the attacker's action. Specifically, $F(\pi'|\pi, u)$ represents the probability that the victim updates its policy from π to π' in the environment changed by u.
- $c : X \times U \times X \rightarrow \mathbb{R}$ is a function denoting the attack cost. It jointly reflects the attack performance (i.e., difference between the victim's policy and the attacker-desired one) and the attack efforts (i.e., aggregate changes to the victim's environment). γ is a discount factor.

Note that we use i to represent the index of an attack epoch in which the victim learns its policy for some episodes.

3.2 Attack Problem Formulation

The attacker aims to learn an attack strategy $\sigma(u|x)$ that achieves policy compulsion on the victim via minimal changes to the victim's training environment. In detail, the attack objective is to minimize the deviation between the victim's policy and the attacker-desired one, while at the same time minimizing the deviation between the poisoned environment dynamics and the natural ones. Therefore, the attack optimization problem is to minimize the cumulative attack costs, which is denoted as:

$$\begin{aligned} &\min_{\sigma} \sum_{i=1}^{\infty} \gamma^i c_i \\ &s.t. \\ c_i &:= \Delta(P_i(s', a'|s, a) || P^*(s', a'|s, a)) \\ P_i(s', a'|s, a) &= T_{e_i}(s'|s, a)\pi_i(a'|s') \\ P^*(s', a'|s, a) &= T_{e_0}(s'|s, a)\pi^*(a'|s'), \end{aligned} \tag{1}$$

where c_i is the attack cost at the i^{th} attack epoch. P_i is a stochastic process [23] over state-action pairs, where the victim follows the policy $\pi_i(a|s)$ in the environment e_i which has been modified by a sequence of tweaks $u_{1:i}$. Similarly, P^* represents an ideal stochastic process, where the victim adopts the attacker-desired policy $\pi^*(a|s)$ in the natural environment e_0. Referring to the definition of stochastic processes P_i and P^*, $\Delta(P_i||P^*)$ measures the deviation jointly caused by the victim's actual policy π_i and its underlying environment dynamics T_{e_i}. We can see that $\Delta(P_i||P^*)$ mathematically describes the attack cost.

Unfortunately, solving the attack optimization problem is intractable in complex or real-world RL systems due to challenges in computing $\Delta(P_i||P^*)$. These challenges are mainly caused by (1) the RL agent's learning algorithm and policy model π are generally black-box to the attacker in the real world; (2) the environment-dynamics model T is typically unknown in most RL tasks, such as control tasks in continuous state domains; (3) even though the environment hyper-parameters can be accessed, it is hopeless to obtain prior knowledge about how the environment hyper-parameters e determine the transition dynamics T_e.

In the following, we propose an approach that simultaneously learns how to infer internal information of an RL system, how to approximate our attack objective, and how to learn an optimal strategy addressing the attack optimization problem. We make minimal assumptions on the attacker's prior knowledge, resulting in a more realistic and scalable attack approach to complex RL tasks.

4 METHOD

This section presents our attack approach and introduces how we tackle the challenges caused by the limited prior knowledge. We start by describing the attack procedure in the double-black-box setting. We then consider how to infer the internal information of the RL system, and finally learn an adaptive environment-poisoning strategy based on an approximation of our attack objective.

4.1 Attack Procedure

To lure an RL agent's policy learning into a desired direction, the attacker poisons the training-environment hyper-parameters according to the information about the agent's behaviour policy and environment dynamics. Unfortunately, such information is hidden from the attacker, resulting in challenges for designing and deploying an adaptive attack. To learn an adaptive environment-poisoning strategy, we incorporate an inference module into the attack framework, which can infer the features of the victim's behaviour and environment during its learning progress.

As shown in the part (a) of Figure 2, an encoder and an attack strategy are essential components of the attack procedure. The encoder captures the feature of the victim's learning progress, and the attack strategy poisons the victim's environment based on the inferred feature. In detail, the attack procedure consists of two stages at each attack epoch i. First, given an observation of the victim's trajectories, the attacker uses the encoder to infer a joint feature z_{i-1} of the victim's policy and its training environment. Second, conditioning on the inferred feature z_{i-1}, the attacker conducts a poisoning action u_i on the environment hyper-parameter e_{i-1},

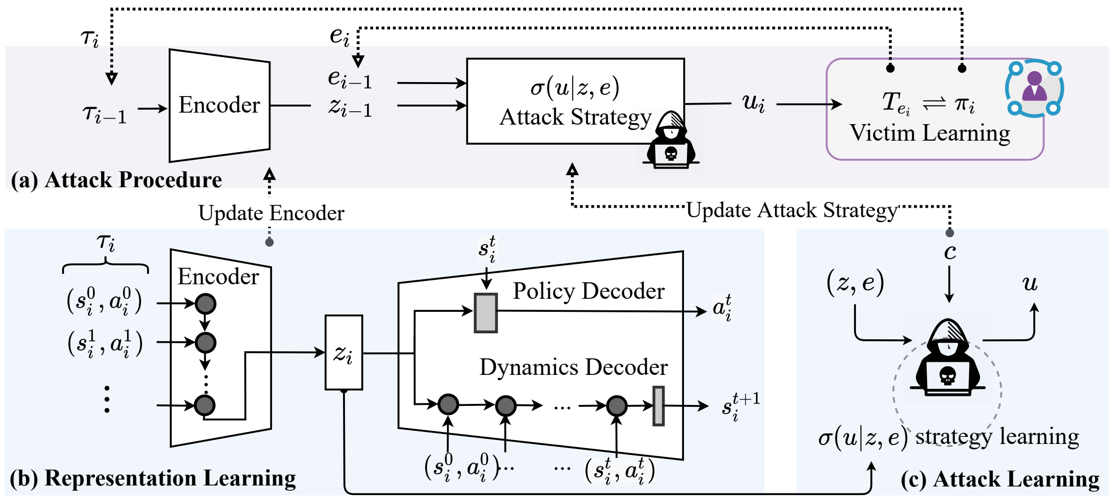


Figure 2: Illustration of double-black-box environment-poisoning attacks: (a) shows the attack procedure; (b) and (c) describe the latent representation learning and attack strategy learning, respectively. The solid line denotes data transfer and the dotted line represents data update.

resulting in poisoned dynamics T_{e_i}. Thereby, the victim, which explores in the training environment T_{e_i}, evaluates and adjusts its policy π_i according to the feedback from the poisoned environment. Such an attack procedure continues until the victim acquires the attacker-desired policy.

4.2 Attack Learning Methodology

As the encoder and the strategy are learned simultaneously, the learning of the double-black-box attack is composed by 1) latent representation learning and 2) attack strategy learning, as shown in the part (b) and the part (c) of Figure 2.

4.2.1 Latent Representation Learning: Without access to the internal information of the RL victim's system, the attacker can only observe the victim's trajectory $\tau = \{s^0, a^0, s^1, a^1, \dots s^t, a^t\}$. Based on the trajectory τ_i collected at the i^{th} attack epoch, the attacker learns a latent embedding z_i that best represents the victim's stochastic process $P_i(s', a'|s, a) = T_{e_i}(s'|s, a) \times \pi_i(a'|s')$. Similarly, τ^* refers to the attacker-desired trajectory, which is used to learn z^* that is the latent representation of the desired stochastic process $P^*(s', a'|s, a) = T_{e_0}(s'|s, a) \times \pi^*(a'|s')$. Note that z reflects the joint information of the victim's environment dynamics and its behaviour policy. As for the effect of environment hyper-parameters (i.e., causal mechanism), it is implicitly expressed in the dynamics.

To learn the latent representation, we design an Encoder-Dual-Decoder network consisting of one encoder network $\mathcal{E}$ and two decoder networks $\{\mathcal{D}^\pi, \mathcal{D}^T\}$, shown as the part (b) of Figure 2. Specifically, the encoder $\mathcal{E}$ approximates $f_\eta(z_i|\tau_i)$ which learns a latent embedding z_i based on the victim's trajectory τ_i. $\mathcal{E}$ is designed as a Long Short Term Memory (LSTM) network to learn long-term dependencies in the trajectory. To learn the encoder $\mathcal{E}$, we interpret the embedding z_i via two decoders: a policy decoder $\mathcal{D}^\pi$ and a dynamics decoder $\mathcal{D}^T$. Here, the policy decoder $\mathcal{D}^\pi$ is a multilayer perceptron (MLP) network. $\mathcal{D}^\pi$ learns $f_\theta(a_i^t|s_i^t, z_i)$ which maps the victim's state s_i^t and the embedding z_i to the distribution over the victim's actions a_i^t. And the dynamics decoder $\mathcal{D}^T$ is a LSTM network that approximates $f_\phi(s_i^{t+1}|s_i^t, a_i^t, z_i)$. It predicts the next state s_i^{t+1} on the condition of the embedding z_i and the state-action pair $\{s_i^t, a_i^t\}$.

Given a collection of the victim's trajectories, we learn the parameter η of the encoder $\mathcal{E}$ and parameters θ, ϕ of the dual decoder $\{\mathcal{D}^\pi, \mathcal{D}^T\}$, by maximizing the following negative cross-entropy objective:

$$\mathbb{E}_{\substack{\tau_1 \sim \mathcal{T}, \\ \tau_2 \sim \mathcal{T} \setminus \tau_1}} \left[\sum_{\langle s,a,s',a' \rangle \sim \tau_2} \log f_\theta(a'|s', f_\eta(\tau_1)) + \log f_\phi(s'|s, a, f_\eta(\tau_1)) \right], \tag{2}$$

where $\mathcal{T}$ is a collection of trajectories at one attack epoch. τ_1 and τ_2 are two different trajectories from the collection $\mathcal{T}$, or from the set $\mathcal{T}^*$ of attacker-desired trajectories.

The encoder $\mathcal{E}$ is particularly important in this work, which is used to produce a latent embedding z given a sequence of state-action pairs. The property of the latent representation makes our attack approach independent of the environment type, i.e., our DBB-EPA can be generally applied to both discrete and continuous environments.

4.2.2 Attack Strategy Learning: Referring to Section 3.2, $\Delta(P_i||P^*)$ mathematically defines our attack cost, and its computation is the key challenge in learning an optimal attack strategy within the double-black-box setting. To solve the challenge, we design an approximation of the attack cost in this section.

Depending on the encoder $\mathcal{E}$, the victim's stochastic process $P_i(s', a'|s, a)$ and the ideal one $P^*(s', a'|s, a)$ can be represented by the latent embedding z_i and z^*, respectively. Thus, we approximate $\Delta(P_i||P^*)$ as $\Delta(z_i||z^*)$, and use the Cosine Similarity [4] to measure the distance between z_i and z^* in the latent space:

$$\Delta(P_i||P^*) := \Delta(z_i||z^*) = 1 - \frac{z_i \cdot z^*}{\|z_i\| \|z^*\|}. \tag{3}$$

Algorithm 1 Learning of DBB-EPA Strategy

```
1: for n_episode = 1,2,... do
2:     Reset victim's embedding $z_0$ and environment $e_0$
3:     for attack epoch $i$=1,2,... do
4:         $u_i \leftarrow \sigma(z_{i-1}, e_{i-1})$    ▷ choose attack action conditioning on victim's policy and environment
5:         $T_{e_i} \leftarrow T_{e_{i-1}}(u_i)$    ▷ alter environment hyper-parameters to poison transition dynamics
6:         observe trajectory $\tau_i$ when victim learns $\pi_i$ in $T_{e_i}$
7:         update encoder $\mathcal{E}$ and dual-decoder $\mathcal{D}^{\pi}$, $\mathcal{D}^{T}$ with $\tau_i$
8:         $z_i \leftarrow \mathcal{E}(\tau_i)$, $z^* \leftarrow \mathcal{E}(\tau^*)$    ▷ infer latent representations from trajectories
9:         $c_i \leftarrow (1-\omega) \times \Delta(z_i||z^*) + \omega \times \Delta(e_i||e_0)$    ▷ approximate the attack cost
10:        update attack strategy $\sigma(u|z,e)$ using samples $(\{z_{i-1}, e_{i-1}\}, u_i, \{z_i, e_i\}, c_i) \in \mathcal{B}$    ▷ $\mathcal{B}$ is a replay buffer
11:        if $\tau_i == \tau_*$ then attack done, go to the next episode
12:        end if
13:    end for
14: end for
```

Since z_i is inferred from the victim's experienced trajectories, $\Delta(z_i||z^*)$ only captures the environment changes that have affected the victim's behaviour, rather than measures the aggregate changes across the entire environment (i.e., attack effort). Considering minimizing the attack effort, we measure the aggregate environment changes $\Delta(e_i||e_0)$ using the normalized euclidean distance between the environment hyper-parameter e_i and the natural one e_0.

Thereby, the approximation of the attack cost c_i is a combination of $\Delta(z_i||z^*)$ and $\Delta(e_i||e_0)$, which is denoted as:

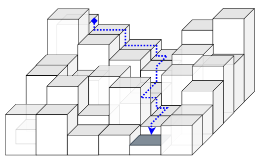
$$\begin{aligned} c_i &= (1-\omega) \times \Delta(z_i||z^*) + \omega \times \Delta(e_i||e_0) \\ &= (1-\omega) \times \left(1 - \frac{z_i \cdot z^*}{||z_i||||z^*||}\right) + \omega \times \frac{||e_i - e_0||_2}{||e_{limit} - e_0||_2}, \end{aligned} \tag{4}$$

where $\omega \in [0, 1]$ is the weight parameter and e_{limit} indicates the limit value of environment hyper-parameters.

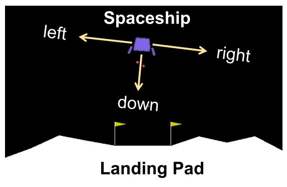
In summary, DBB-EPA uses an encoder to obtain latent representations of an RL system's internal information, and learns an adaptive attack strategy based on an approximated attack cost. The learning of DBB-EPA strategy is finalized as Algorithm 1. Due to the representation and the approximation in the latent space, DBB-EPA is applicable in poisoning an RL agent in both discrete and continuous environments, as empirically discussed in the next section.

5 EXPERIMENT

In this section, we first evaluate DBB-EPA on a tabular-RL agent in a didactic grid-world task, showing that DBB-EPA achieves effective training-time attack with minimal prior knowledge of the RL system. Then, we show the feasibility and the scalability of DBB-EPA in more complex RL tasks. Implementation codes can be found at https://github.com/JoanaHXU/DBB-EPA.



(a) 3D grid world



(b) LunarLander

Figure 3: Illustration of experiment environments

5.1 Discrete State Domains

We evaluate DBB-EPA on a tabular-RL agent performing a navigation task in a 3D grid world [23, 34] where the environment-dynamics model is unknown. The purpose of this experiment is to evaluate the design of the inference module (as the part (b) in Figure 2), and to discuss the approximation of the attack cost (as Equation 4).

5.1.1 Experiment Setting. As shown in Figure 3a, the 3D grid world simulates mountains or rugged terrain, which serves as the stochastic environment for a navigation task.

Attack Settings. In the 3D grid world, the success of moving from one cell to the neighboring one is proportional to their relative elevation, as a result, changing elevation is a mechanism to modify the environment dynamics. Thus, we consider the *elevation* as the environment hyper-parameter which can be manipulated by the attacker. Additionally, the attack objective is to stealthily force the victim to reach the destination along the boarder of the grid world, instead of following the optimal path (i.e., the blue line in Figure 3a). Since the state domain and the action domain in the 3D grid world are discrete and countable, the attacker-desired policy can be defined manually.

Implementation and Measurement. We adopt Deep Deterministic Policy Gradient (DDPG) [16] to learn the attack strategy. We measure the attack performance using *attack success rate* [34] at each episode during the victim's learning process. Here, *Attack success rate* is the percentage of the attacker-desired states that have been attack successfully, specifically, successful attack is the one in which the victim performs the attacker-desired action in the desired state.

5.1.2 Results and Discussion.

Attack Performance Evaluation. We evaluate DBB-EPA performances in comparison with the white-box attack (i.e., TEPA) [34]. All the results are generated by the attack strategies which are learned and evaluated on a Q-learning RL agent. As shown in Figure 4, attack success rate of DBB-EPA is comparable to that of the white-box attack. Consequently, this result suggests that our proposed attack is capable of poisoning a black-box RL agent's

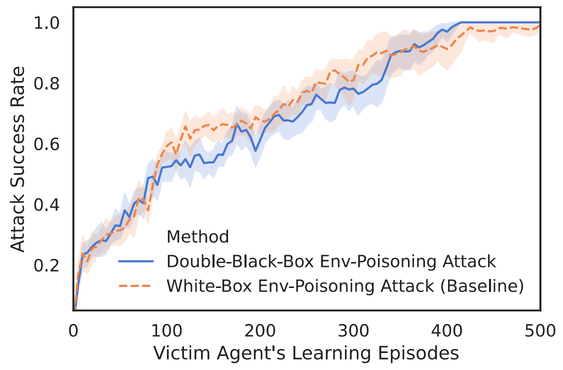


Figure 4: Performance of double-black-box attack in comparison with white-box attack in 3D Grid World

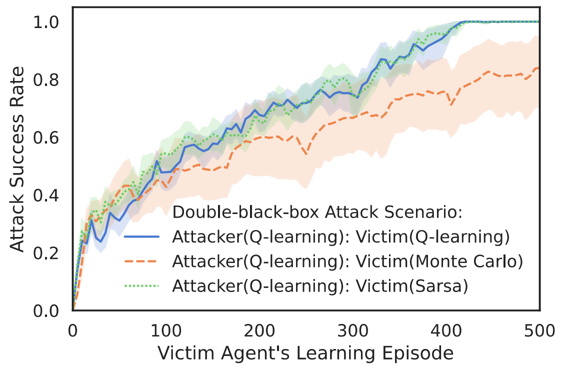


Figure 5: Performance of double-black-box attack against different RL agents in 3D Grid World

Table 1: Deviations between manipulated environment hyper-parameters with the natural ones, under different settings of weight parameters. Deviation is measured when attack success rate reaches 100%**.**

ω	Deviation (L_2)	Deviation Percentage
0.0	21.55 ± 1.77	0.538 ± 0.044
0.1	17.88 ± 1.82	0.447 ± 0.045
0.2	17.61 ± 1.90	0.440 ± 0.047
0.3	16.49 ± 1.29	0.412 ± 0.032
0.4	16.43 ± 1.08	0.411 ± 0.027
0.5	16.11 ± 2.02	0.402 ± 0.050

policy in an unknown environment. The result further indicates the effectiveness of the proposed inference module and the designed attack-cost approximation.

Transferability Evaluation. To evaluate the transferability of the DBB-EPA strategy, we learn an attack strategy based on a black-box Q-learning proxy agent, and deploy the strategy on three kinds of victim agents of which learning algorithms include Q-learning, Sarsa and Monte Carlo [32]. As shown in Figure 5, the Q-learning and Sarsa victims obtain the attacker-desired policy within 500 learning episodes; while MC victim reaches 80% success rate at 500^{th} episode and needs more time to acquire the attacker-desired policy. We see that the efficiency of the transferred attack against the MC-victim is somewhat lower than that against the Q/Sarsa victim. This result can be explained by the characteristics of victims' learning algorithms. Specifically, Sarsa and Q-learning are temporal-difference (TD) algorithms while MC is an on-policy experience-based learning algorithm. TD learns the policy online at every time step, whereas MC has to wait until the end of the episode for the policy to be updated. In practice, it appears that TD learning works more efficiently than MC, which affects the efficiency of learning the attacker-desired policy. In summary, our DBB-EPA strategy can be successfully transferred among agents which use different learning algorithms, while the attack efficiency is influenced by the characteristics of the victim's RL learning algorithm.

Analysis of Weight Parameter ω. When designing the attack-cost approximation as Equation 4, we explicitly consider the aggregate environment changes with a weight parameter ω. Here, we empirically show the effect of ω on the learned attack strategy. Table 1 displays deviations between the poisoned environment and the natural one, when the attack success rate has reached 100%. Deviations of environment hyper-parameters are measured by the L_2 norm, and they are also represented by the percentage of hyper-parameters that have been altered. As shown in Table 1, the environment deviation decreases as ω increases. Consequently, the explicit representation of environmental deviations contributes positively to the control of attack efforts. However, according to our observation, it is becoming increasingly difficult to train a successful attack strategy with the increasing ω. For example, the training with $\omega = 0.1$ converges within 300 episodes while the training with $\omega = 0.5$ takes 800 episodes to converge. Therefore, learning an attack strategy involves a tradeoff. The attack strategy which requires fewer attack efforts (i.e. changes in the environment) is more difficult to learn.

Discussions on Computational Time. We present the computational time introduced by the inference module (i.e., Encoder-Dual-Decoder network) on GPU (i.e., Nvidia RTX A6000) in terms of the attack learning and deployment. First, the learning time of the DBB-EPA strategy is 83,066 seconds, which is 1.19 times more than that of the white-box TEPA strategy (i.e., 69,537 seconds). The 20% increase is acceptable for learning the internal information of RL systems. Second, the deployment time of the DBB-EPA attack and the white-box TEPA attack are 30.09 seconds and 29.53 seconds, respectively. The impact of Encoder on the deployment time can be negligible (about 2% increase) due to its small network size and small computation load.

5.2 Continuous State Domains

Experimental results in the discrete environment have demonstrated the effectiveness of our inference module and attack-cost approximation. In this part, we further evaluate the feasibility and the scalability of DBB-EPA in more complex RL tasks. Note that the complexity of RL tasks is interpreted from two perspectives: (1) the complexity of the RL agent is scaled by shifting from tabular-RL algorithms up to deep-RL algorithms; (2) the complexity of the environment is scaled from discrete state domains up to continuous domains. In the following, we show the DBB-EPA performance against DRL agents in LunarLander.

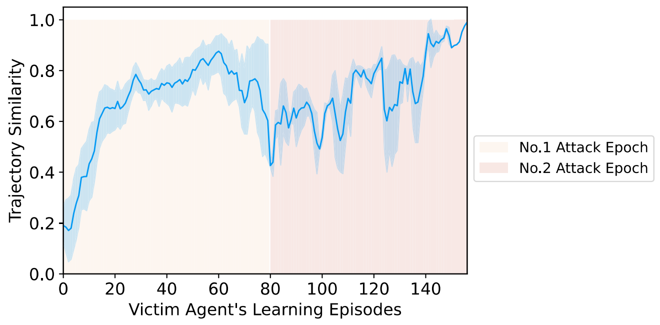


Figure 6: Attack evaluation on a DQN victim in LunarLaner

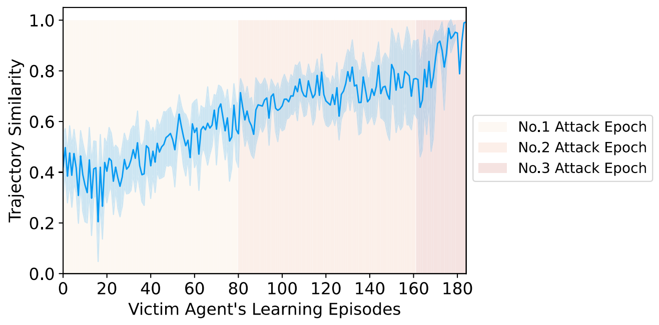
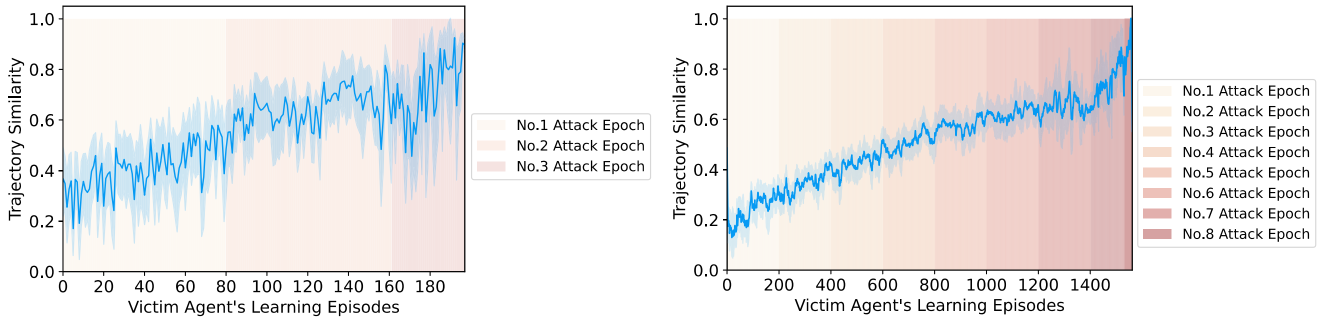
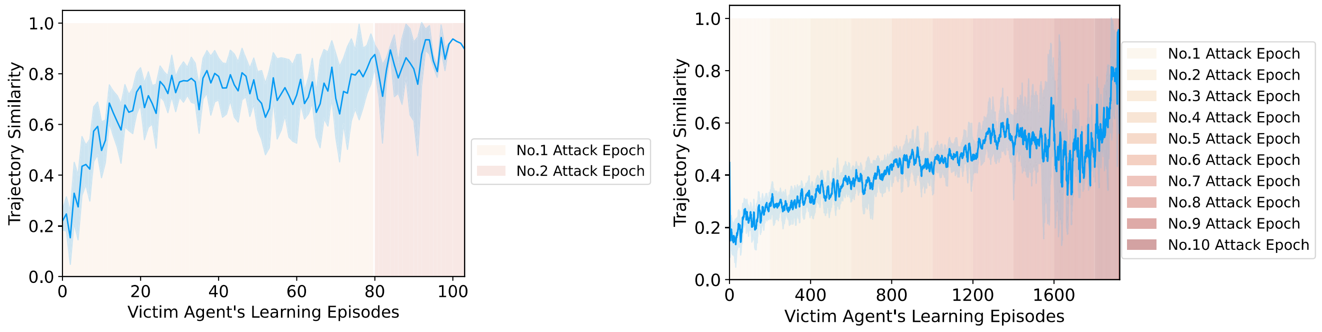


Figure 7: Attack evaluation on a PPO victim in LunarLaner



(a) Transfered attack on a *PPO* victim **(b) Transfered attack on a *VPG* victim**

Figure 8: Transferability evaluation of an attack strategy learned from a black-box *DQN* proxy agent in LunarLander



(a) Transfered attack on a *DQN* victim **(b) Transfered attack on a *VPG* victim**

Figure 9: Transferability evaluation of an attack strategy learned from a black-box *PPO* proxy agent in LunarLander

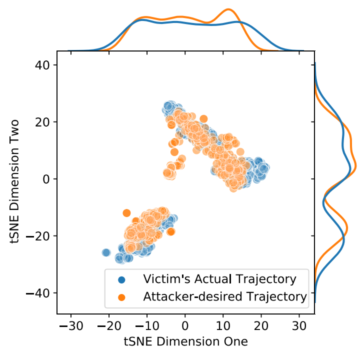
5.2.1 Experiment Settings. LunarLander [2] is a simulation environment for the trajectory optimization problem, where the environment state is continuous and the dynamics model is unknown. As shown in Figure 3b, the task is to safely land a spaceship between the flags smoothly.

Attack Settings. In light of the observation that there is random wind which might influence the spaceship's trajectory, we choose *wind* as the environment hyper-parameter that can be implemented by the attacker. Specifically, the wind power affects the spaceship's moving distance, and the wind direction influences the spaceship's moving direction. For example, the spaceship is increasingly difficult to moving downward with increasing upward wind power. Thus, the *wind* implementation is a mechanism to affect the environment dynamics. Additionally, the attacker aims to prevent the spaceship from landing on the pad while keep the spaceship safe. In detail, the attacker-desired trajectory is that the spaceship hovers around the right flag for at least 1000 time steps. We obtain such an attacker-desired policy using the reward-shaping solution [35].

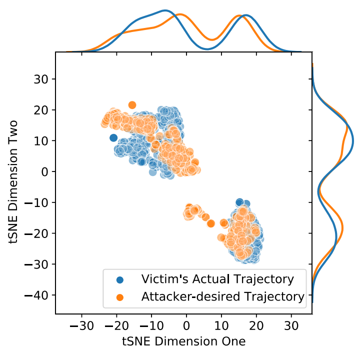
Implementation and Measurement. We adopt Twin Delayed DDPG (TD3) [7] to learn the DBB-EPA strategy. The attack strategy is evaluated on black-box DRL agents of which the learning algorithms include an off-policy algorithm Deep Q-Network (DQN) [18], on-policy algorithms Vanilla Policy Gradient (VPG) [32] and Proximal Policy Optimization (PPO) [28]. Additionally, the attack performance is measured by the trajectory similarity, i.e., the Cosine Similarity between the latent representations. If the trajectory similarity is more than 90%, we consider that the victim's policy has been successfully poisoned.

5.2.2 Results and Discussion.

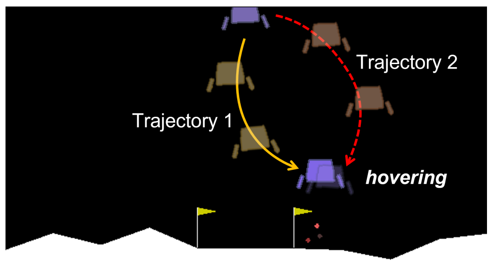
Attack Performance Evaluation. We learn an attack strategy based on a DQN proxy agent and a PPO proxy agent, respectively. The



(a) tSNE visualization for a DQN victim



(b) tSNE visualization for a PPO victim



(c) Illustration of attacker-desired trajectories

Figure 10: Visualization of the victim's poisoned policies and the attacker-desired trajectories

attack strategy is deployed at every 80 episodes during the victim's learning process, and the attack performance is measured at the end of each learning episode. As shown in Figure 6, the similarity between the DQN victim's actual trajectory with the attacker-desired one reaches nearly 100% within two attack epochs. It indicates that the DQN victim's policy have been poisoned successfully by the attack strategy which is learned on the DQN proxy. When attacking PPO victim using the strategy learned on the PPO proxy, Figure 7 shows that trajectory similarity reaches above 90% within three attack epochs. These results indicate that DBB-EPA is successful in forcing an DRL agent to learn the attacker-desired trajectory in a continuous environment, without prior knowledge of both the agent's algorithm and the training environment.

Transferability Evaluation. We evaluate the transferability of DBB-EPA strategy in the LunarLander setting. The attack strategies are trained based on a DQN proxy agent and a PPO proxy agent, respectively. Afterwards, they are applied to attack victims adopting different learning algorithms. Figure 8 and Figure 9 show the performance of the transferred attack strategies. Overall, the DBB-EPA strategies successfully induce the DRL victim to learn the attacker-desired trajectory regardless of the victim's learning algorithm. In detail, as shown in Figure 8, the PPO victim's policy is poisoned within 3 attack epochs while VPG victim requires 10 epochs for being attacked successfully, with the strategy learned on a DQN proxy. As shown in Figure 9, with the strategy learned on a PPO proxy, DQN victim is successfully attacked within 2 attack epochs whereas VPG victim's policy is manipulated using 10 epochs. In summary, these observations indicate that an DBB-EPA strategy, which is learned on a on-policy DRL agent, can be successfully transferred to poison a off-policy DRL agent, and vice versa.

Furthermore, we notice that the attack efficiency varies for different victims, which is attributable to the characteristics of the victim's learning algorithm. Specifically, the off-policy DQN algorithm learns its policy at each timestep using samples in a replay buffer. The on-policy PPO algorithm updates its policy with each sample of a trajectory at the end of the episode. The VPG algorithm, on the other hand, updates its policy based on one return of an entire trajectory at each episode. Due to the different frequency of policy updating, VPG's policy learning process is slower than that of DQN or PPO. This explains why VPG victim requires more attack epochs to obtain the target policy. In summary, the DBB-EPA strategy can be transferred to poison different DRL agents' policies, nevertheless, the attack efficiency depends on the characteristics of the victim's specific learning algorithm. This conclusion is consistent with the transferability discussion for tabular RL in the discrete domain (as Section 5.1.2).

Visualization of Poisoned Policy. To examine the similarity between the victim's poisoned trajectory with the attacker-desired one, we use tSNE [33] to visualize their representations in the latent space. We collect 500 trajectories generated by the victim's poisoned policy and the attacker-desired policy, respectively. Individual trajectory is encoded as an embedding and visualized by tSNE. In Figure 10a and 10b, the victim's poisoned trajectories are clustered same as the attacker-desired ones, which has been shown by the overlapping distribution curves along the two tSNE dimensions. Notice that two clusters of embeddings exists, which represent the two kinds of attacker-desired trajectories as shown in Figure 10c, i.e., the spaceship experiences two kinds of trajectories and finally keeps hovering around the right flag more than 1000 time steps. In conclusion, both the DQN agent and the PPO agent have been misled to learn the attacker-desired policies in the poisoned training environment, and both experience the attacker-desired trajectories in the natural testing environment.

6 CONCLUSION

In this paper, we propose an environment-poisoning attack against an RL agent at training time, with minimal prior knowledge of the RL system. Assuming only the ability to alter the environment hyper-parameters, our attack achieves minimal and adaptive environment poisoning, forcing a *black-box* RL agent to learn an attacker-desired policy in an *unknown* environment. It achieves comparable performance to that of the white-box attack in the grid world, and succeeds in poisoning the DRL agent's policy in continuous environments. In summary, our study investigates the security threat posed by environment hyper-parameters, which can serve as a test-bed core to analyze vulnerabilities of RL to training-environment poisoning.

ACKNOWLEDGMENTS

This research was in part supported by the NTU SUG "Choice manipulation and Security Games".

REFERENCES

[1] Xiaoxuan Bai, Wenjia Niu, Jiqiang Liu, Xu Gao, Yingxiao Xiang, and Jingjing Liu. 2018. Adversarial Examples Construction Towards White-box Q Table Variation in DQN Pathfinding Training. In *Proceedings of the 3rd International Conference on Data Science in Cyberspace*. IEEE, Guangzhou, China, 781–787.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:1606.01540

[3] Tong Chen, Jiqiang Liu, Yingxiao Xiang, Wenjia Niu, Endong Tong, and Zhen Han. 2019. Adversarial Attack and Defense in Reinforcement Learning-from AI Security View. *Cybersecurity* 2, 1 (2019), 1–22.

[4] Kenneth Ward Church. 2017. Word2Vec. *Natural Language Engineering* 23, 1 (2017), 155–162.

[5] Antonio Coronato, Muddasar Naeem, Giuseppe De Pietro, and Giovanni Paragliola. 2020. Reinforcement Learning for Intelligent Healthcare Applications: A Survey. *Artificial Intelligence in Medicine* 109 (2020), 101964–101964.

[6] Niloufar Eghbali, Tuka Alhanai, and Mohammad M Ghassemi. 2021. Patient-Specific Sedation Management via Deep Reinforcement Learning. *Frontiers in Digital Health* 3 (2021), 1–9.

[7] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Stockholmsmässan, Stockholm Sweden, 1587–1596.

[8] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep Reinforcement Learning That Matters. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*. AAAI, Louisiana, USA, 3207–3214.

[9] Yunhan Huang and Quanyan Zhu. 2019. Deceptive Reinforcement Learning under Adversarial Manipulations on Cost Signals. In *Proceedings of the International Conference on Decision and Game Theory for Security*. Springer, Stockholm, Sweden, 217–237.

[10] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Huang, and Dusit Niyato. 2021. Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. *IEEE Transactions on Artificial Intelligence* 1, 1 (2021), 1–21.

[11] Taylor W Killian, George Konidaris, and Finale Doshi-Velez. 2017. Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*. AAAI, California, USA, 4949–4950.

[12] Sunyong Kim and Hyuk Lim. 2018. Reinforcement Learning based Energy Management Algorithm for Smart Energy Buildings. *Energies* 11, 8 (2018), 2010–2029.

[13] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems* 1, 1 (2021), 1–18.

[14] Sangyoon Lee and Dae-Hyun Choi. 2022. Federated Reinforcement Learning for Energy Management of Multiple Smart Homes with Distributed Energy Resources. *IEEE Transactions on Industrial Informatics* 18, 1 (2022), 488–497.

[15] Roman Liessner, Jakob Schmitt, Ansgar Dietermann, and Bernard Bäker. 2019. Hyperparameter Optimization for Deep Reinforcement Learning in Vehicle Energy Management. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence*. SCITEPRESS, Prague, Czech Republic, 134–144.

[16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous Control with Deep Reinforcement Learning. In *Proceedings of 4th International Conference on Learning Representation*. ICLR, San Juan, Puerto Rico, 1–14.

[17] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. 2019. Policy Poisoning in Batch Reinforcement Learning and Control. In *Proceedings of the 33th Conference on Neural Information Processing Systems*. ACM, Vancouver, Canada, 14570–14580.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. In *Proceedings of the 27th Conference on Neural Information Processing Systems*. ACM, USA, 1–9.

[19] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. 2018. Learning to Adapt in Dynamic, Real-world Environments through Meta-Reinforcement Learning. In *Proceedings of 6th International Conference on Learning Representations*. ICLR, Vancouver, Canada, 1–17.

[20] Xinlei Pan, Weiyao Wang, Xiaoshuai Zhang, Bo Li, Jinfeng Yi, and Dawn Song. 2019. How You Act Tells a Lot: Privacy-leaking Attack on Deep Reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. IFAAMS, Auckland, New Zealand, 368–376.

[21] Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. 2017. Virtual to Real Reinforcement Learning for Autonomous Driving. In *Proceedings of 28th British Machine Vision Conference*. BMVA, London, British, 1–13.

[22] Christian Perez, Felipe Petroski Such, and Theofanis Karaletsos. 2020. Generalized Hidden Parameter MDPs: Transferable Model-based RL in a Handful of Trials. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI, New York, USA, 5403–5411.

[23] Zinovi Rabinovich, Lachlan Dufton, Kate Larson, and Nick Jennings. 2010. Cultivating Desired Behaviour: Policy Teaching via Environment-dynamics Tweaks. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems*. IFAAMS, Toronto, Canada, 1097–1104.

[24] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2020. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 7974–7984.

[25] Amin Rakhsha, Xuezhou Zhang, Xiaojin Zhu, and Adish Singla. 2021. Reward Poisoning in Reinforcement Learning: Attacks Against Unknown Learners in Unknown Environments. In *Proceedings of the 35th Conference on Neural Information Processing Systems*. ACM, Online, 1–22.

[26] Elsa Riachi, Muhammad Mamdani, Michael Fralick, and Frank Rudzicz. 2021. Challenges for Reinforcement Learning in Healthcare. arXiv:2103.05612 [cs.LG]

[27] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep Reinforcement Learning Framework for Autonomous Driving. *Electronic Imaging* 2017, 19 (2017), 70–76.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[29] Tomah Sogabe, Dinesh Bahadur Malla, Shota Takayama, Seiichi Shin, Katsuyoshi Sakamoto, Koichi Yamaguchi, Thakur Praveen Singh, Masaru Sogabe, Tomohiro Hirata, and Yoshitaka Okada. 2018. Smart Grid Optimization by Deep Reinforcement Learning over Discrete and Continuous Action Space. In *Proceedings of the 7th World Conference on Photovoltaic Energy Conversion*. IEEE, Hawaii, USA, 3794–3796.

[30] Sumedh A Sontakke, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. 2021. Causal Curiosity: RL Agents Discovering Self-Supervised Experiments for Causal Representation Learning. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Online, 9848–9858.

[31] Yanchao Sun, Da Huo, and Furong Huang. 2021. Vulnerability-Aware Poisoning Mechanism for Online RL with Unknown Dynamics. In *Proceedings of 10th International Conference on Learning Representation*. ICLR, Vienna, Austria, 1–27.

[32] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press, Cambridge, Massachusetts, USA.

[33] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605.

[34] Hang Xu, Rundong Wang, Lev Raizman, and Zinovi Rabinovich. 2021. Transferable Environment Poisoning: Training-time Attack on Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. IFAAMAS, Online, 1398–1406.

[35] Haoqi Zhang, David C Parkes, and Yiling Chen. 2009. Policy Teaching through Reward Function Learning. In *Proceedings of the 10th ACM Conference on Electronic Commerce*. Association for Computing Machinery, New York, NY, United States, California, USA, 295–304.

[36] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. 2020. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 11225–11234.