Transferable Environment Poisoning: Training-time Attack on Reinforcement Learning

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

Rundong Wang
Nanyang Technological University
Singapore
rundong001@e.ntu.edu.sg

Lev Raizman
University of Waterloo
Waterloo, Canada
lev.raizman@uwaterloo.ca

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

ABSTRACT

Studying adversarial attacks on Reinforcement Learning (RL) agents has become a key aspect of developing robust, RL-based solutions. Test-time attacks, which target the post-learning performance of an RL agent's policy, have been well studied in both white- and black-box settings. More recently, however, state-of-the-art works have shifted to investigate training-time attacks on RL agents, i.e., forcing the learning process towards a target policy designed by the attacker. Alas, these SOTA works continue to rely on white-box settings and/or use a reward-poisoning approach. In contrast, this paper studies environment-dynamics poisoning attacks at training time. Furthermore, while environment-dynamics poisoning presumes a transfer-learning capable agent, it also allows us to expand our approach to black-box attacks. Our overall framework, inspired by hierarchical RL, seeks the minimal environment-dynamics manipulation that will prompt the momentary policy of the agent to change in a desired manner. We show the attack efficiency by comparing it with the reward-poisoning approach, and empirically demonstrate the transferability of the environment-poisoning attack strategy. Finally, we seek to exploit the transferability of the attack strategy to handle black-box settings.

KEYWORDS

Reinforcement Learning; Environment Poisoning; Security

ACM Reference Format:
Hang Xu, Rundong Wang, Lev Raizman, and Zinovi Rabinovich. 2021. Transferable Environment Poisoning: Training-time Attack on Reinforcement Learning. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Proliferation of applied Reinforcement Learning (RL) techniques has drawn the ever increasing attention to their safety and robustness. In fact, the study of RL's susceptibility to subversion has become a prominent research area of its own. The majority of existing works [7, 8, 10, 11, 23] focus on test-time attacks against RL agents, challenging an already learned and fixed policy. These works successfully attack and degrade the performance of a deployed, RL-generated policy, though the policy itself is not subject to the methods' manipulation. However, a research sub-stream that aims to subvert an RL agent's policy as it is being generated, during training time, has been forming [2, 12, 20, 27]. Admittedly, these methods are successful to a degree, and propel the RL agent towards a target policy, designed and chosen by the attacker. However, several strong assumptions wrt access to the RL agent's perceptions and knowledge of the agent's processes (white-box assumption) greatly limit the applicability of these approaches.

In more detail, current training-time attack methods assume that the attacker has full knowledge of the RL agent's learning algorithm and policy model. Furthermore, it is assumed that the agent's perceptions and memory are accessible, so that the complete interaction information between an agent and its environment is manipulable. Thus, reward manipulation/poisoning is a common weapon of choice to interfere with the agent's learning process and drive it towards the target policy. However, the agent's perceived rewards are determined by both external feedback and the agent's *intrinsic* environment [22]. For example, in a classical story "Phil prepares his breakfast" [1, 22], Phil's chosen action might receive different rewards depending on his hunger level, his mood and other features of his body, which are internal and private properties of the agent. Similarly, some robots' perceived rewards are determined by the configuration of embedded sensors, encapsulated away from clients/attackers. In such scenarios, reward manipulation [25–27] is infeasible, as there is no access to the manner in which rewards are generated. Similar failure easily befalls approaches that work by poisoning the RL agent's memory [12]. Thus, it is necessary to design an attack approach that works only through features and properties of the environment that are external to the RL agent, i.e., environment's response to actions, dynamics. Additionally, even though we may have some design documentation on the agent's learning algorithm, it is important to design an attack method that will be as independent from such information as possible. In other words, the method and its attack strategy should operate under the black-box assumption and be transferable across different RL agents. In this paper, we develop such an approach.

Specifically, we develop a method to automatically design a transferable attack strategy, trained in a white-box setting with a proxy agent and then applied to an unknown (black-box) victim agent. In more detail, we deploy a bi-level Markov Decision Process

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

(MDP) framework as a means of finding a training-time attack strategy by environment-dynamics poisoning. At a lower-level, an RL proxy agent explores an MDP environment, seeking to maximize its expected cumulative rewards. At a higher-level, the attacker seeks to tweak the dynamics of the lower-level MDP to drive the proxy agent towards a target behaviour/policy. We assume that the tweak can be generated at regular intervals with respect to the lower-level time-line, and that prior to the tweak the higher-level attacker obtains a reliable estimate of the proxy agent's current policy. While the attacker's main goal is to drive the proxy agent towards acquiring the target policy, the attacker is also limited in its ability to change the proxy agent's environment. In fact, we relate the degree to which the environment has been changed to the attacker's *effort* and *stealth*. Thus, we build the attack optimization around the combined deviation of a) the environment dynamics from their natural form *and* b) the proxy agent's policy from the target policy. The combined deviation measurement is based on the Kullback-Leibler Divergence Rate [19] between two autonomic systems.

We demonstrate the effectiveness of the proposed method in two stages. First, we develop it to handle white-box scenarios, where the victim agent's algorithm is known to the attacker; hence, the proxy agent is precise in simulating the victim of the attack. We then show, how the approach can be extended to black-box scenarios, where the proxy agent approximates the victim using the latter's observed interaction trajectories. Furthermore, we explore the transferability of the environment-poisoning attack strategy between RL agents by explicitly creating scenarios where the proxy agent and the victim run different and distinct algorithms. Overall, the contribution of this paper can be summarized as follows:

- We propose a transferable environment-dynamics poisoning attack (TEPA) against an RL agent at training time;
- We present a close-loop attack design framework based on a bi-level Markov Decision Process architecture;
- We show that the bi-level architecture can be successfully resolved using a Deep RL technique that generates valid attack strategies;
- We demonstrate that our approach generates effective white-box and black-box attacks, and investigate the transferability of the environment-poisoning solution.

2 RELATED WORK

In this section, we review existing works on training-time attacks against RL and present how this work is different from them.

Training-time Attacks against RL. Different from test-time attacks, which perturb the performance of the well-trained policy, training-time attacks aim to enforce an arbitrary target policy on the RL agent. Existing works mostly poison the agent's policy by manipulating rewards at training time [12, 27]. Here, [12] poisons rewards in the training data set for batch RL, and [27] studies adaptive reward-poisoning attacks on a Q-learning RL agent. Of particular interest is the work proposed in [20], which poisons environment transition dynamics. Its target victim is an RL agent using a minimization-regret framework as the learning algorithm, whose objective is to maximize average rewards in undiscounted infinite-horizon settings. Thus, this environment-poisoning attack in [20] is suitable for cyclic tasks which have no termination state. In contrast, our proposed environment-poisoning attack works in episodic MDP and targets an RL agent whose objective is to maximize its cumulative rewards. Additionally, unlike [12, 20, 27] that focus on a specific RL agent, our method is not constrained by how the agent learns the policy.

White-box and Black-box Attacks. For test-time attacks on RL agents, existing works have investigated both white-box [10, 11, 23] and black-box settings [7, 8]. In white-box settings, the attacker is assumed to have full knowledge of the victim's learning algorithm and policy model. During the deployment of the victim's policy, the attacker perturbs the victim's perceived states in order to change its decision on selecting actions. In the works [7, 8] which study black-box attacks, the attacker has no access to the RL agent's model weights. These works are developed based on a key concept: transferability of adversarial examples (i.e., crafted malicious inputs) [15]. Here, the concept 'transferability' means that adversarial examples, which can mislead one model, are also able to perturb other models. Transferability has been demonstrated effective for models whose architectures or training sets are different, as long as these models are trained for the same task. Inspired by the transferability of adversarial examples, [7, 8] learn attacks on a proxy model and build black-box adversarial examples to fool deep RL agents. However, in the area of training-time attacks on RL, to our knowledge, no approach is feasible in black-box settings. In this work, we aim to explore the effectiveness of our method for attacking black-box RL agents at training time.

Offline and Online Attacks. In existing training-time attack works, there are mainly two settings: offline and online attacks. For offline attacks, the attacker has full knowledge of the training data set. It makes one decision on rewards manipulation, and then provides the poisoned training data set to the RL agent for its policy planning [12, 20]. On the other hand, online attacks mean that the attacker sequentially manipulates the feedback signal when interacting with the RL agent [20, 27]. It requires the attacker to access the agent's transitions (i.e., current state, chosen action, next state and reward), to make attack decisions on-the-fly. Such attacks are generally adaptive to the victim's learning progress, however, they are constrained to white-box settings. Different from offline and online attacks, we propose an intermediate attack framework where the environment is sequentially poisoned at regular interval during the victim's learning process. The attacker makes an attack decision in response to the victim's policy features, without needing to grasp every transitions of the victim. The proposed setting allows us to extend training-time attacks to black-box settings where learning algorithms or policy models are unknown.

Environment Poisoning and Shaping. It has been shown in previous works that RL agents are very sensitive to training environments [6]. Thus, manipulating environment is an effective way to influence the agent's policy learning [9, 17, 18, 20, 21]. Generally, these works can be grouped as either environment shaping or environment poisoning, based on their objectives. Specifically, environment shaping aims to speed up the agent acquiring its own optimal policy, which is popular in behaviour teaching [9, 17, 21]. Environment poisoning, on the other hand, enforces an arbitrary

target policy on the agent. The target policy can be a suboptimal one for the agent. It is usually studied in adversarial attack fields [18, 20]. Even though the objectives are different, teaching and attacking are mathematically equivalent. This means our work can be used to teaching a specific target policy. Furthermore, when our work is discussed in the teaching area, it is important to note the difference between our work and inverse RL (IRL) [3, 16] due to their similar objectives. IRL performs teaching by human inputs, and requires the agent to be able to estimate the teacher's motivation; while our work depends on automated environment design and performs effectively even when the agent has no intention of learning.

3 PRELIMINARIES

The development of our work is inspired by Behaviour Cultivation (BC) [18]. In this section, we describe the core idea of BC and how it fits into our proposed attack method.

Behaviour Cultivation: The goal of BC [18] is to induce an RL agent to learn a desired behaviour with minimized changes of environment dynamics. In BC, the learner has no intention of following the desired behaviour, instead, it seeks to maximize its own benefit. The teacher aims to induce the learner to acquire an arbitrary behaviour that is desired by the teacher. These characteristics allow BC to be effective in adversarial attacks on an RL agent's policy. In this work, we borrow the core idea of enforcing an arbitrary desired policy via environment-dynamics tweaks from BC. We extend this core idea into a closed-loop control sequence of environment-dynamics modifications responding to the learner's actual learning progress.

Since BC aims to achieve a desired behaviour on the learner via minimized environment-dynamics changes, it considers the cost of changing dynamics when minimizing the deviation between the learner's actual policy with the desired one. BC uses Kullback Leibler Divergnece Rate (KLR) [19] as the measurement of this deviation. In this work, we also adopt KLR to measure the attack cost to achieve attack success with control of the attack effort.

Kull-Leibler Divergence Rate: Kullback-Leibler divergence measures how one probability distribution is different from a reference probability distribution.

Definition.1: Given discrete probability distributions p and q, the Kullback-Leibler divergence of q from p is

$$D^{KL}(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}$$

When Kullback-Leibler divergence is extended to Markovian processes, it is called Kullback-Leibler Divergence Rate [19].

Definition.2: Given Markov Processes X_t^1 and X_t^2, the Kullback-Leibler Divergence Rate (KLR) is

$$D_{KLR}(X^1||X^2) = \lim_{n\to\infty} \frac{1}{n} D^{KL}(P(X^1 = x_n)||P(X^2 = x_n))$$

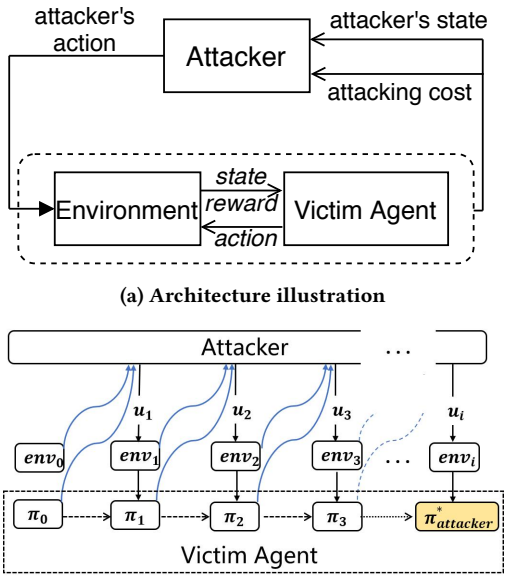
When the Markov Process is described by two conditional transition matrices, $P(x'|x)$ and $Q(x'|x)$, the KLR is proven to be

$$D_{KLR}(P||Q) = \sum_x D^{KL}(P(x'|x)||Q(x'|x)) \times p(x) \qquad (1)$$

where $p(x)$ is the stationary distribution of P [19].

4 PROBLEM STATEMENT

In this section, we formulate our environment-poisoning attack problem using a bi-level MDP architecture, which is shown in Figure 1a. Here, the victim is an RL agent who interacts with the environment, seeking maximized cumulative rewards. The attacker is also an RL agent who regards the victim as a dynamic system. As shown in Figure 1b, at intervals of the victim's learning process, the attacker manipulates the victim's environment dynamics, responding to the victim's momentary policy information. The attacker's objective is to acquire an attack strategy which succeeds in policy poisoning with minimized changes of environment.



(a) Architecture illustration

(b) Attack Deployment

Figure 1: Environment-Poisoning Attack in Bi-level MDPs

Victim-level MDP. The victim's Markovian environment is represented by the tuple $< S, A, T_{\hat{u}}, r, q_0, \gamma >$. Here, $s \in S$ is an environment state and $q_0(s)$ is a distribution over initial states. A is the set of the victim's available actions and $r : S \times A \times S \to \mathbb{R}$ is the reward function. $\gamma \in (0, 1)$ is the discount factor. $T_{\hat{u}} : S \times A \times S \to [0, 1]$ is the probabilistic state transition function with tweaks of hyperparameters coming from space $\hat{U}$ (described in the attack-level MDP). Thus, $T_{\hat{u}}(s'|s, a)$ represents the probability of the environment state changing from s to s', given that the environment dynamics was tweaked by $\hat{u} \in \hat{U}$ and the victim chose the action a. The victim's objective is to find an optimal policy $\pi(a|s)$ which maximizes the cumulative discounted rewards.

Attack-level MDP. We define the attacker's environment as a higher-level Markovian process, which is denoted as the tuple $< \Theta, U, F, \gamma_a, c, \theta^* >$ where:

- Θ is the attacker's state space. It is the set of the victim's policy parameterisations. θ^* denotes the target policy designed by the attacker.

- U is the attacker's action space. $u_i \in U$ is the tweak to the victim's environment hyper-parameters. Here, the hyper-parameters control how the environment responds to the victim's action. u_0 means that environment hyper-parameters are not changed by the attacker. $u_{1:i}$ represents the aggregate outcome of a sequence of tweaks, which is also termed as $\hat{u_i}$ within confines of a set $\hat{U}$.
- F is the attacker's environment transition probability function. It captures how the victim updates its policy. $F(\theta'|\theta, \hat{u})$ represents the probability that the victim changes its policy parameter from θ to θ' in the environment tweaked by $\hat{u}$.
- γ_a is the discount factor.
- $c : \Theta \times \hat{U} \rightarrow \mathbb{R}$ is a function representing the attack cost. It denotes the combined impact of the victim's (possibly undesirably) parameterised policies and the attacker's aggregate tweaks in the victim's environment.

The attacker aims to minimize the cumulative discounted attack cost $C = \sum_{i=1}^{\infty} \gamma_a^i c_i$, where i is the attack epoch. The attacker's objective is to find an attack strategy $\sigma(u_i|\theta_{i-1}, u_{1:i-1})$ which can stealthily force the victim to acquire the target policy while minimizing cumulative changes to the environment dynamics.

5 METHODOLOGY

In this section, we first design the attack-level optimality criteria in white-box settings, and then investigate black-box attacks.

5.1 White-box Attacks

The white-box attack setting is common in research on adversarial attack [7, 12, 20, 27]. In white-box settings, the attacker is assumed to know the victim's learning algorithm and its policy model. Before describing the attack optimality criteria, we first describe the target policy which is designed by the attacker. Given a target state set $S^* \subseteq S$, the target policy is denoted as:

$$\pi_{\theta^*}(s) = \begin{cases} a^* & \text{s} \in S^* \\ \pi_{\theta_i}(s) & \text{s} \notin S^* \end{cases} \tag{2}$$

where a^* is the target action desired by the attacker. $\pi_{\theta_i}(s)$ is the victim's actual policy. $\pi_{\theta^*}(s)$ is a partial target policy which is more applicable for large-size discrete or continuous state domains, in comparison with the complete target policy (i.e., define desired actions in *all* the states) proposed in BC [18].

Based on the definition of $\pi_{\theta^*}(s)$, we measure the attack cost c_i at each attack epoch i, as described by the following:

$$\begin{aligned} c_i(\theta, u) &= D_{KLR}(P_i||P^*) \\ &s.t. \\ D_{KLR}(P_i||P^*) &= \sum_{s,a} q_i(s)\pi_{\theta_i}(a|s)D_i^{KL}(s,a) \\ D_i^{KL}(s,a) &= \sum_{s',a'} P_i(s',a'|s,a)\log\frac{P_i(s',a'|s,a)}{P^*(s',a'|s,a)} \\ P_i(s',a'|s,a) &= T_{u_{1:i}}(s'|s,a)\pi_{\theta_i}(a'|s') \\ P^*(s',a'|s,a) &= T_{u_0}(s'|s,a)\pi_{\theta^*}(a'|s') \\ q_i(s') &= \sum_{s,a} q_i(s)\pi_{\theta_i}(a|s)T_{u_{1:i}}(s'|s,a) \end{aligned} \tag{3}$$

Here, $q_i(s)$ is the stationary distribution of the victim's environment MDP. $P_i(s', a'|s, a)$ represents a stochastic process over state-action pairs, where the victim follows the policy $\pi_{\theta_i}(a|s)$ in the environment modified by a sequence of tweaks $u_{1:i}$, where $u_j \in U$ for all $j \in [1 : i]$. Similarly, $P^*(s', a'|s, a)$ is the ideal Markovian process from the perspective of the attacker.

Note that Equation 3 defines attack cost computation at each attack epoch i. The attacker and the victim have different time scales. At each attack epoch i, the environment transition dynamics $T_{u_{1:i}}$ is fixed. The victim interacts with the poisoned environment within t_{max} time steps and finds policy π_{θ_i}. Then, the attacker recognizes the parameterisation of the victim's updated policy and accordingly tweaks environment hyper-parameters, starting another attack epoch $i + 1$.

Based on the proposed optimality critieria, the training of the attack strategy is described in Algorithm 1. Here, line 9 ∼ 10 introduce how the attacker poisons the environment dynamics, and line 11 means that the victim updates its policy in the poisoned environment using algorithm f_{victim}. In line 12 ∼ 13, the attack cost is computed following Equation 3. Then, the attacker's transitions are saved in the replay buffer D for optimizing the strategy network. When the victim chooses desired actions in all the target states, the attack is done and the training goes to the next episode.

Algorithm 1 Training of Attack Strategy in White-box Settings

1: **Input**:
   T_0: default environment dynamics
   π_0: victim's initial policy
   π_{θ^*}: target policy designed by attacker
2: **Output**:
   σ: attack strategy network
3: $P^* \leftarrow \pi_{\theta^*} \times T_0$: compute ideal Markovian process
4: **for all** *episode_num* **do**
5: $T_0 = T_{u_0}$: reset default environment dynamics
6: $\pi_0 = \pi_{\theta_0}$: revert victim to initial policy
7: $x_0 \leftarrow \{\pi_0, u_0\}$: attacker's input
8: **for** i=1 to I_{max} **do**
9: $u_i \leftarrow \sigma(x_{i-1})$: choose attack action
10: $T_i \leftarrow T_{u_{1:i}}$: poison environment dynamics
11: $\pi_i \leftarrow f_{victim}(\pi_{i-1}, T_i)$: obtain victim's updated policy
12: $P_i \leftarrow \pi_i \times T_i$: compute actual Markovian process
13: $c_i \leftarrow D_{KLR}(P_i, P^*)$: compute attack cost
14: $x_i \leftarrow \{\pi_i, u_{1:i}\}$: update attacker's input
15: $D \leftarrow \{x_{i-1}, u_i, x_i, c_i\}$: save transitions to replay buffer
16: Update attack strategy network σ
17: **if** $\{\pi_i(s) == \pi_{\theta^*}(s)|s \in S^*\}$ **then**
18: $attackDone = 1$: go to the next episode
19: **end if**
20: **end for**
21: **end for**

We use Deep RL algorithms to learn the attack strategy. We choose Deep Q-learning (DQN) [13] in discrete attack action domain, and adopt Twin Delayed DDPG (TD3) [4] for continuous attack actions.

5.2 Black-Box Attacks

In this section, we will introduce how to apply the proposed bi-level environment-poisoning attack in black-box settings. We study black-box attacks in two settings. First, the attacker has no prior information about the victim agent's learning algorithm, i.e., how the agent learns the policy. The attacker is assumed to know the victim agent's policy model, i.e., what the agent's momentary policy is. In the second setting, the attacker knows nothing about the victim agent's learning algorithm as well as its policy. Instead, the attacker can only observe the agent's behaviour trajectories. These two settings are closer to real-world attack scenarios than the setting in white-box training-time attacks [12, 20, 27]. To our knowledge, there is no prior work investigating black-box attacks on RL agents at training time.

5.2.1 Transferable Environment-Poisoning Attack (TEPA). In the first setting, the challenge for the attacker is in designing its attack strategy without information about the victim's learning algorithm. To address this issue, we would like to demonstrate *transferability* of poisoned training environments. As [22], in comparison with supervised learning, the characteristic feature of RL is that the agent learns optimal actions via feedback from the training environment, rather than being directly instructed by correct actions. Thus, the interaction with the training environment plays a crucial role in determining the RL agent's optimal policy. As for the agent's learning algorithms or policy representations, they only affect how efficiently the agent finds the optimal policy, instead of determining what the optimal policy is. Therefore, the training environment which poisons one agent's policy can be transferred to attack other RL agents, even if these agents utilize different learning algorithms or policy representations. We term this property as *transferability* of poisoned training environments.

We utilize transferability of poisoned training environments to solve the challenge in the first black-box setting. Here, we propose that the attacker learns its attack strategy on a white-box proxy agent (i.e., learning algorithm is known by the attacker), and then transfers the strategy to attack a black-box victim agent.

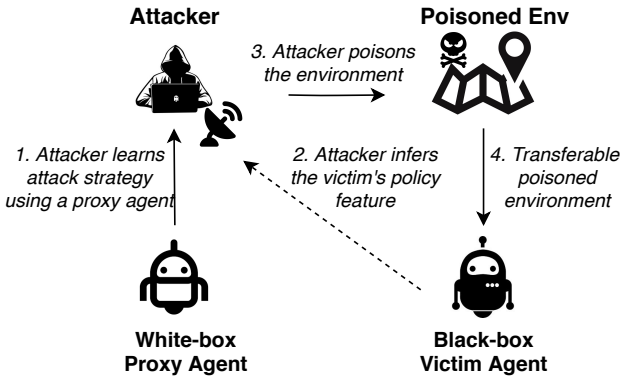


Figure 2: Black-box, transferable poisoning attacks

5.2.2 TEPA with Victim Modeling. Since it is difficult for the attacker to have prior knowledge on the victim's private information in real-world scenarios, the second black-box setting is more realistic and worth being investigated. Based on the solution in the first setting, the key task for the attacker is to infer the victim's policy when deploying responsive attacks.

Inspired by generative policy representations [5, 24], we use the encoder-decoder neural network to obtain policy parameterisations which can best represent the victim's policy features. Specifically, at each attack epoch i, the attacker first collects the victim's successive N state-action transitions $\mathbf{X^i} = \{s_t^i, a_t^i\}$ derived from the victim's policy π_{θ_i}, where $t \in [t_k, t_{k+N})$. Then, the attacker learns a representation function $f_\omega : \mathbf{X^i} \to \mathbb{R}^d$ which encodes trajectories $\mathbf{X^i}$ as a d-dimensional real-valued vector embedding. This embedding represents the victim's policy features. The decoder network is a policy function $f_\phi : S \times A \times \mathbb{R}^d \to [0, 1]$ which maps the victim's state and the embedding to the distribution over its actions. The parameter ω and ϕ in the encoder-decoder neural network are learned via maximizing the negative cross-entropy objectives:

$$\mathbb{E}_{\mathbf{x_1^i} \sim \mathbf{X^i}, \mathbf{x_2^i} \sim \mathbf{X^i}} \left[\sum_{\langle s^i, a^i \rangle \sim \mathbf{x_2^i}} \log f_\phi(a^i | s^i, f_\omega(\mathbf{x_1^i})) \right] \tag{4}$$

Here, $\mathbf{x_1^i}$ and $\mathbf{x_2^i}$ are two different trajectories from the transition set $\mathbf{X^i}$, which are used to train the encoder and the decoder network, respectively. Since the output of the encoder network (i.e., embedding) represents the victim's policy features, it is used as the input to the attacker.

In summary, as described in Figure 2, the attacker learns the attack strategy on a white-box proxy agent, and infers the black-box victim agent's policy features according to trajectory observations. Then, it applies the transferable attack strategy to the black-box victim, responding to the victim's policy features.

6 EXPERIMENT

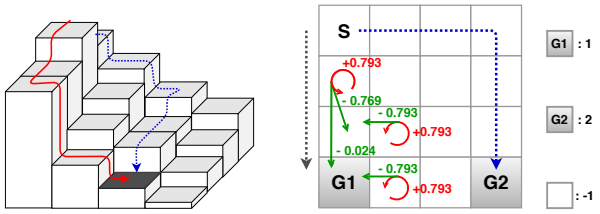
In this section, we present experiments to show the attack performance of bi-level environment(env)-poisoning attack under white-box and black-box settings.

6.1 Experiment Settings

6.1.1 Experiment Environment. Robot navigation is one fundamental problem in robotics and commonly used to test RL algorithms [14]. Therefore, we choose to attack an RL agent performing navigation tasks in Grid World environment.

3D Grid World. The 3D grid world is proposed in [18] to simulate a mountain or rugged terrain. As shown in Figure 3a, there are associated elevations among cells of the 3D grid world. Success of moving from one cell to the neighboring one is proportional to the relative elevation between these two cells. Thus, changing elevation can affect how the environment responds to the agent's action, which is a mechanism to modify the environment dynamics. The 3D grid world is the victim agent's environment. The agent's aim is to find an optimal path from the start cell to the destination. At each time step, the agent can move in one of the four cardinal directions with the reward -1. For the attacker, the attack objective is to stealthily force the agent to follow a target path reaching the destination. As shown in Figure 3a, the blue line represents the

victim agent's optimal path in the natural elevation setting, whereas the red line is the target path designed by the attacker.



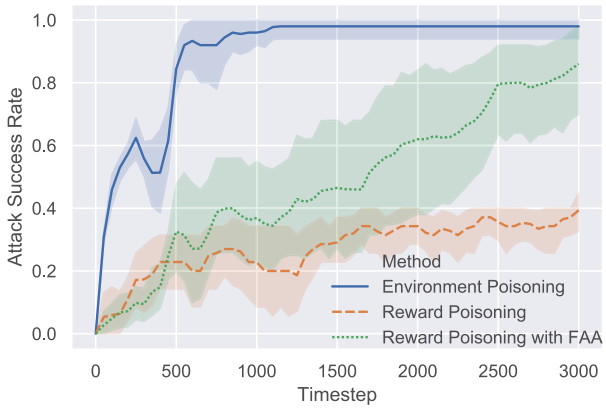
(a) 3D Grid World (b) 2-Goal Grid World

Figure 3: Discrete domains: (a) In the 3D grid world, the shallow cell is the goal state. (b) In the 2-Goal grid world, there are two goal states G_1 and G_2

Grid World with Two Goal States: Inspired by the experiment design in [12], we extend the 3D grid world with two goal states G_1 and G_2 shown in Figure 3b. We set the default elevation of each cell to be the same. In this domain, the agent is rewarded -1 for each step until it reaches $G1$ or $G2$ with reward +1 and +2, respectively. The agent's task is to reach one of goal states with maximized cumulative rewards. The agent will reach G_1 following its own optimal policy. However, the attacker aims to induce the agent to reach the other goal state, $G2$, by changing cell elevations. In Figure 3b, the blue line is the target path while the black line is the agent's optimal option. In addition, the red circle and green arrow represent how the state transition probabilities are changed by the attacker. For example, the red circle with value +0.793 means the probability of staying still has been increased by 0.793 when the agent choose a specific action.

6.1.2 Implementation. As introduced in Section 5.1, the attacker learns the env-poisoning attack strategy using Deep RL learning algorithms, such as DQN [13] or TD3 [4]. The policy network is represented by a multi-layer neural network as INPUT(80)-FC(400)-ReLU-FC(300)-ReLU-FC(16). When attacking victims in the 3D grid world, the attacker manipulates the cell elevation continuously and uses TD3 to learn the attack strategy. For attacks in the 2-Goal grid world, the attacker chooses DQN to train its policy network with a discrete attack action set. Additionally, victim agents may choose Q-learning, Sarsa and Monto Carlo (MC) [22] as learning algorithms with parameter configurations: $\epsilon = 0.1, \gamma = 1.0, \alpha = 0.1$.

6.1.3 Measurement. The attack is regarded as successful if the victim agent chooses the desired action a^* in target states s^*. We define *attack success rate* to denote the percentage of the target state set that has been attacked successfully. In this experiment, we measure the attack performance using the *attack success rate* during the victim agent's learning process. The efficiency of an attack strategy is measured by its convergence rate, with faster convergence meaning higher attacking efficiency. Furthermore, a successful transferable strategy is identified as one with more than 80% success rate for attacking a victim agent when the attack strategy is trained on a white-box proxy agent.



(a) 3D Grid World

(b) 2-Goal Grid World

Figure 4: Attack performance of Env-Poisoning Attack in comparison with Reward-Poisoning Attack

6.1.4 Baseline. We use one state-of-the-art approach, adaptive reward-poisoning attack [27], as our baseline. The adaptive reward-poisoning attack shares the same attack objective as ours: to force an RL agent to acquire a target policy designed by the attacker. In the adaptive reward-poisoning attack, the attacker is formulated as an RL agent and uses TD3 [4] to train the attack strategy. The attacker poisons the victim's policy by manipulating its rewards on-the-fly. There are two kinds of reward-poisoning methods proposed in [27]. The first one, *reward poisoning*, regards the victim's transition and Q tabular $< s, a, s', r, Q >$ as the input to the attacker. The other one, *reward poisoning + FAA*, combines a fast adaptive attack (FAA) algorithm into the reward poisoning. In *reward poisoning + FAA*, the target states are ranked in descending order by their distance to the starting states, and are attacked individually at each time. Furthermore, the attacker forces the victim to fix the Q value $Q[s^*]$ once the target action is achieved in the target state s^*. As such, adaptive reward-poisoning attack controls the victim agent's learning algorithm, which limits its potential applications. Instead, our env-poisoning attack strategy poisons the training environment to affect the victim's learning process, without control on the victim, leading to broader potential applications.

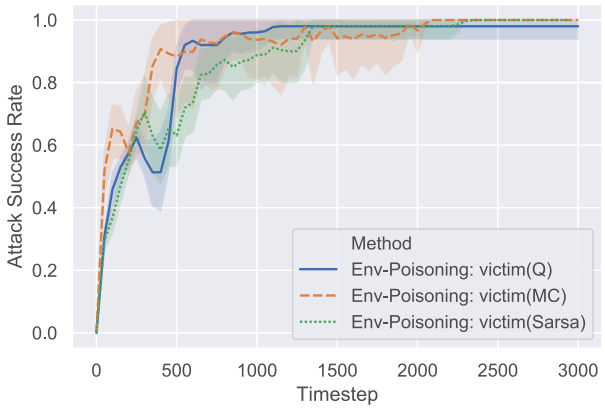


Figure 5: Attack effectiveness against a range of RL agents

6.2 Experiment Results and Discussion

To avoid any confusion, we discuss the experiment results using the Question-and-Answer format. Question 1 and 2 address the performance comparison with the state-of-the-art training-time attack methods. In Question 3 and 4, we show the transferability of the env-poisoning attack strategy for victim agents. Lastly, in Question 5, the effectiveness of env-poisoning attack in black-box settings is discussed in detail.

6.2.1 White-box Attack Settings. Under the white-box attack settings, the victim's learning algorithm and policy representation are known to the attacker.

***Question.1**: How is the performance of Env-Poisoning Attack in comparison with Reward-Poisoning Attack?*
The attack success rate during the victim's learning process is shown in Figure 4. The env-poisoning strategy shows a better attack success rate in both 3D grid word and 2-Goal grid world. In Figure 4a, we observe a fast increasing attack success rate during the victim's learning progress comparing with *reward poisoning* and *reward poisoning + FFA*. When the victim's learning process finishes, our env-poisoning strategy shows 95% attack success rate comparing with 85% *reward poisoning + FFA* and 40% *reward poisoning*. For the 2-Goal grid world, our env-poisoning strategy achieves 94% attack success rate, better than the *reward poisoning + FFA* and *reward poisoning*. Note that env-poisoning attack will not control the victim's learning way, leading to more practical applications in real world.

***Question.2:** Is the Bi-level Env-Poisoning Attack framework effective for various RL agents?*
Existing works on training-time attacks are mostly designed for a specific RL algorithm, such as batch RL [12] and Q learning [27]. To show that the env-poisoning attack is not constrained by types of victim's learning algorithms, we present the attack results against RL agents with three learning algorithms respectively: Q-learning, Sarsa and MC. As shown in Figure 5, we observe that all the attacks achieve about 100% success rate within 2000 timesteps in the victim's learning process. It indicates the proposed attack is general to RL agents without limitation on victim's learning algorithms.

6.2.2 Black-box Attack Settings. Under the black-box attack settings, the victim's learning algorithm and policy representation are unknown to the attacker.

***Question.3:** Is the Env-Poisoning Attack strategy transferable among various RL agents?*
In Section 5.2, we propose that the env-poisoning attack can be transferred from white-box settings to black-box settings. Here, we evaluate this idea under the first black-box setting without leaking the victim's learning algorithm to the attacker. When training attack strategies, we develop three white-box proxy agents: Q-learning, Sarsa and MC. There are three attack strategies learned under these proxy agents, respectively. The strategies are then transferred to attack victims with arbitrary learning algorithms. Figure 6 shows performance of each attack strategy on different victims. Overall, we observe that the attack strategy trained on a white-box proxy agent is effective for attacking a victim agent with a different learning algorithm. All these strategies achieve more than 80% attack success rate within 3000 time steps. This shows that env-poisoning attack strategies are successfully transferable among RL agents which adopt different learning algorithms. Thus, due to the transferability, our proposed that attack can be applicable to the black-box setting where the victim's learning algorithm is unknown.

***Question.4:** How does the proxy agent affect the transferability of the Env-Poisoning Attack strategy?*
When comparing each plot in Figure 6, we observe that these three attack strategies vary in attack performance. Specifically, the strategy trained on the Q-learning proxy agent (as in Figure 6a) performs slightly better than the one trained on Sarsa (as in Figure 6b), while it is much better than the strategy trained on MC (as in Figure 6c). These results suggest that the type of the proxy agent has effect on the transferable performance of the env-poisoning attack strategy.

So, the question is what kind of learning algorithm should be chosen for proxy agents? Here, MC is a on-policy experience-based learning algorithm whereas Sarsa and Q-learning are on-policy and off-policy temporal-difference algorithms, respectively. Based on the characteristics of each learning algorithm, the difference of transferable performance is due, in part, to the exploration level of proxy agents. In on-policy MC, the policy is determined by experienced trajectories; while in off-policy Q-learning, the agents use ϵ-greedy action selection that causes more uncertainty of the policy and leads to more exploration in the training environment. Therefore, we should choose a proxy agent which performs enough exploration when training a transferable attack strategy.

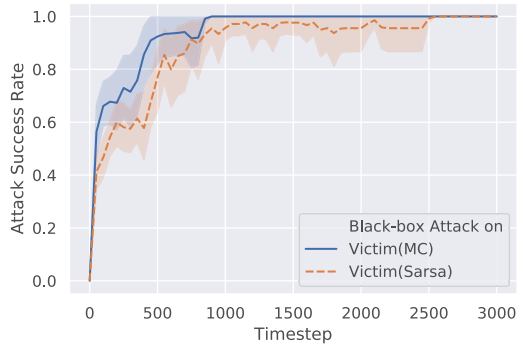
***Question.5:** Can the proposed Env-Poisoning method effectively attack a black-box RL agent at training time?*
The attacker has no prior information about the victim's learning algorithm and policy representations under the black-box setting. As such, there are two challenges: designing a training-time attack strategy and poisoning the environment in response to the victim's policy. In this experiment, we still use a white-box proxy agent to learn a transferable attack strategy, and furthermore adopt a policy modeling approach to infer the victim's policy features from trajectories $< s_t, a_t, s_{t+1}, a_{t+1}, ..., s_{t+N}, a_{t+N} >$. Here, the policy features are learned using an encoder-decoder network, and

(a) Trained on Q-learning proxy agent **(b) Trained on Sarsa proxy agent** **(c) Trained on MC proxy agent**

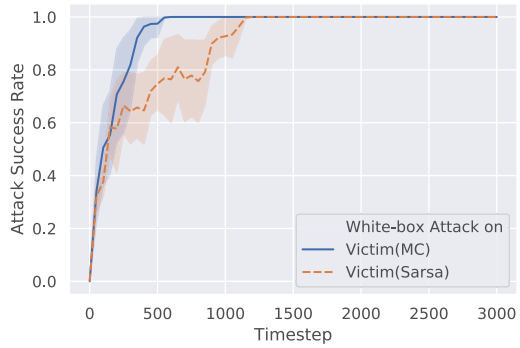
Figure 6: Transferability of the environment-poisoning attacks: proxy-agent and the attacked RL explicitly differ.

Table 1: Architecture of Encoder-Decoder network used for representing policy. ReLU and Layer normalization is applied after each layer except the last one.

Encoder			Decoder	
Type	Input Dim.	Output Dim.	Input Dim.	Output Dim.
Linear	20	36	6	36
Linear	36	36	36	36
Linear	36	5	36	1



(a) Black-box attack



(b) White-box attack

Figure 7: Effectiveness of environment-poisoning attack on black-box agents whose learning algorithm and policy model is unknown.

network configuration is shown in Table 1. The attack strategy is trained using a Q-learning proxy agent and then transferred to attack two black-box victims. To emphasize the transferability, we set the victims using different learning algorithms, Sarsa and MC. As Figure 7 shows, the transferable env-poisoning attack strategy is successful in enforcing desired actions in target states. Its final attack performance is as good as the white-box attack while its success rate grows slowly. This is partly because the attacker needs time to collect the victim's trajectory data and accordingly infers its policy features. This result shows that our proposed method effectively attack an RL agent whose private information (i.e., learning algorithm and policy model) is totally unknown.

7 CONCLUSION

In this work, we present a bi-level environment-poisoning attack framework for an RL agent at training time. The attacker exercises influence on a victim agent's learning experience via manipulating its environment dynamics. The objective of the attacker is to stealthily force the agent to learn a target policy with minimized environment changes. We evaluate the environment-poisoning attack in both white-box and black-box settings. We demonstrate transferability of poisoned training environments, and show that attack strategies can be trained using a white-box proxy agent and transferred to poison a black-box victim's policy. Then, we analyze how the proxy agent affects the transferable attack strategy, and conclude that more exploration by the proxy agent can lead to a strategy with better transferability. Furthermore, we combine generative policy representation with environment-poisoning attacks, and demonstrate our work is effective for attacking a black-box RL agent.

This work automatically generates an adaptive attack on training environments for an RL algorithm, thus can serve as a baseline to evaluate the robustness capabilities of that RL algorithm. While it yields promising results, the experiment domain used here is quite simplistic. The investigation of more sophisticated environments, in which the environment transition-dynamics function is unknown by the attacker, is a topic for on-going research.

ACKNOWLEDGMENTS

This research is supported by the NTU SUG "Choice manipulation and Security Games".

REFERENCES

[1] Philip E Agre. 1988. *The dynamic structure of everyday life.* Technical Report. Massachusetts Institute of Technology Cambridge Artificial Intelligence Lab.

[2] Vahid Behzadan and Arslan Munir. 2017. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In *Proceedings of the 13th International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, New York, USA, 262–275.

[3] Daniel S Brown and Scott Niekum. 2019. Machine Teaching for Inverse Reinforcement Learning: Algorithms and Applications. In *Proceedings of the 33th AAAI Conference on Artificial Intelligence*, Vol. 33. AAAI press, Hawaii, USA, 7749–7758.

[4] 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.

[5] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. 2018. Learning Policy Representations in Multiagent Systems. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Stockholmsmässan, Stockholm Sweden, 1802–1811.

[6] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2017. Deep Reinforcement Learning that Matters. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*. AAAI press, California, USA, 1,26.

[7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. In *Proceedings of the 5th International Conference on Learning Representations (Workshop)*. ICLR, Vancouver, BC, Canada.

[8] Matthew Inkawhich, Yiran Chen, and Hai Li. 2019. Snooping Attacks on Deep Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMS, Auckland, New Zealand, 557–565.

[9] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. 2017. Equi-Reward Utility Maximizing Design in Stochastic Environments. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI, Melbourne, Australia, 4353–4360.

[10] Jernej Kos and Dawn Song. 2017. Delving into Adversarial Attacks on Deep Policies. In *Proceedings of the 5th International Conference on Learning Representations (Workshop)*. ICLR, Vancouver, BC, Canada, 6.

[11] Yenchen Lin, Zhangwei Hong, Yuan-Hong Liao, Mengli Shih, Mingyu Liu, and Min Sun. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI, Melbourne, Australia, 3756–3762.

[12] 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.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529–533.

[14] 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.

[15] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-box Attacks using Adversarial Samples.

[16] Kamalaruban Parameswaran, Rati Devidze, Volkan Cevher, and Adish Singla. 2019. Interactive Teaching Algorithms for Inverse Reinforcement Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI, Macao, China, 2692–2700.

[17] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. 2020. Automatic Curriculum Learning for Deep RL: A Short Survey. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. IJCAI, Yokohama, Japan, 4819–4825.

[18] 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.

[19] Ziad Rached, Fady Alajaji, and L Lorne Campbell. 2004. The Kullback-Leibler Divergence Rate between Markov Sources. *IEEE Transactions on Information Theory* 50, 5 (2004), 917–921.

[20] 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.

[21] Jette Randløv. 2000. Shaping in Reinforcement Learning by Changing the Physics of the Problem. In *Proceedings of the 27th International Conference on Machine Learning*. PMLR, Stanford, CA, USA, 767–774.

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

[23] Edgar Tretschk, Seong Joon Oh, and Mario Fritz. 2018. Sequential Attacks on Agents for Long-Term Adversarial Goals. In *Proceedings of the ACM Computer Science in Cars Symposium*. ACM, Munich, Germany.

[24] Rundong Wang, Runsheng Yu, Bo An, and Zinovi Rabinovich. 2020. I2HRL: Interactive Influence-based Hierarchical Reinforcement Learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. IJCAI, Yokohama, Japan, 3131–3138.

[25] Haoqi Zhang and David C Parkes. 2008. Value-Based Policy Teaching with Active Indirect Elicitation. In *Proceedings of the 23th AAAI Conference on Artificial Intelligence*, Vol. 8. AAAI press, Chicago, Illinois, 208–214.

[26] 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.

[27] 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.