Universal Black-Box Targeted Reward Poisoning Attack Against Online Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

This work proposes the first universal black-box targeted attack against online reinforcement learning through reward poisoning during training time. Our attack is universally efficient against any efficient learning algorithm training in general RL environments and requires limited attack budgets and computational resources. We generalize a common feature of the efficient learning algorithms and assume that such algorithms would mostly take the optimal actions or actions close to them during training. We quantify the efficiency of an attack and propose an attack framework where it is feasible to evaluate the efficiency of any attack instance in the framework based on the assumption. Finally, we find an instance in the framework that requires a minimal per-step perturbation, which we call 'adaptive target attack.' We theoretically analyze and prove a lower bound for the attack efficiency of our attack in the general RL setting. Empirically, on a diverse set of popular DRL environments learned by state-of-the-art DRL algorithms, we verify that our attack efficiently leads the learning agent to various target policies with limited budgets.

1 Introduction

Online deep reinforcement learning (DRL) algorithms have great potential to be applied in industrial applications such as robotics (Christiano et al., 2017), recommendation systems (Afsar et al., 2021), and natural language processing Ouyang et al. (2022). In such applications, the reward signals during training are usually feedback from human users, which raises the threat of training-time reward poisoning attacks. There may exist adversarial human users who deliberately provide malicious rewards. A malicious attack goal can be manipulating the DRL agent to follow certain behaviors during training with malicious properties, such as unsafe or harmful behavior, to benefit the attacker. Such behavior can still lead to high performance according to the agent's metric, making the attack harder to detect. For example, consider the scenario of training a restaurant's recommendation system. A restaurant wants to mislead the recommendation system to recommend itself more likely. For this purpose, the restaurant can deliberately provide false negative feedback for its competitors and positive feedback for itself. Then, during training, a vulnerable recommendation system might mostly recommend the malicious restaurant, which is unfair and harms the benefits of the restaurants, users, and the system. We show the framework of targeted poisoning attacks against online RL in Figure 1. Prior studies on targeted attacks mainly focus on simpler tabular settings (Rakhsha et al., 2020; Xu et al., 2021; Zhang et al., 2020; Banihashem et al., 2022). The results only show the vulnerability of current tabular RL algorithms, and the attacks usually only work in the white-box setting against specific learning algorithms, which may not be practical. In this work, we aim to expose the practical vulnerabilities of state-of-the-art DRL algorithms by finding a targeted attack that works efficiently in a realistic setting.

Universal black-box targeted attack in DRL and the challenges: Here, we discuss the practical constraints on an attack.

1. **Universal black box:** The learning agent can directly defend against white-box attacks by hiding its learning algorithm and training parameters from the public. Given that many different efficient RL algorithms have been developed (van Hasselt et al., 2016; Wang et al., 2016; Haarnoja et al., 2018; Dankwa

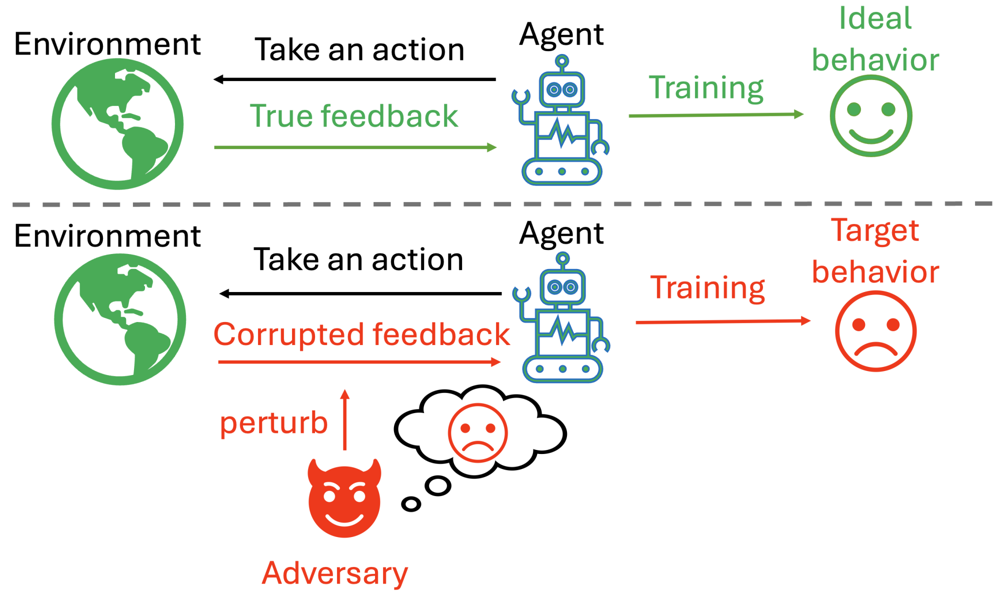

Figure 1: The framework of online reinforcement learning under adversarial poisoning attacks. The learning agent can interact with the environment by taking an action at each timestep. Ideally, when there is no attack, the agent can observe the true feedback from the environment and then learn an ideal behavior desired by the agent. However, in practice, adversaries exist in the environment that can inject corruption into the feedback from the environment. The agent can only train on the corrupted dataset and may learn a target behavior desired by the attacker instead.

& Zheng, 2019), the agent can arbitrarily choose an efficient algorithm. Therefore, a realistic attacker should be oblivious to the learning agent, and its attack strategy should universally work against different algorithms an agent might use.

2. **Real-time with limited computational power:** In online RL, the data collection during the training process happens in real-time. Therefore, a realistic attack should also be able to inject perturbations during training in real-time. In addition, it is more practical to run the attack with limited computational resources, as rich computational resources may not always be available to a malicious attacker.

We do not consider the computational cost of constructing the target policy, as it is an input to the targeted attack problem. In general, the efficiency of an attack is measured by the attack's influence on the learning agent's output with limited perturbations on the training process (attack budgets). We formally show the details of the attack constraints and efficiency in the next section. Developing an efficient and realistic attack is challenging. First, a universal attack should work for different algorithms an agent might use, and it is hard to determine exactly what these algorithms are for an agent. Second, in the black-box setting, the attacker cannot exactly predict an agent's behavior without detailed knowledge of the training setup. To achieve high efficiency, the attack should optimize the budget it needs during the training process while working with limited computational resources. This becomes especially difficult without predicting the agent's behavior under attack.

Our contributions: To the best of our knowledge, our attack is the first universal black-box targeted attack applicable to the DRL setting. The key insight for making the attack universal is a common feature of efficient learning algorithms that we exploit: an efficient DRL algorithm can learn the optimal solution or a similar one through strategic exploration in the environment. Using this feature, we characterize the set of efficient algorithms a rational learning agent might consider. In the universal black box setting, it is hard to approximate the universal behavior of all efficient learning algorithms under an arbitrary attack. We develop an attack framework called 'efficient adversarial reward engineering' such that the behavior of an arbitrary efficient learning algorithm becomes predictable under any attack under the framework. In this framework, the attacker perturbs the feedback signals during training as if they are generated by an adversarial RL environment desired by the attacker. Finally, we find the instance in the framework that requires a minimal attack budget on per-step perturbation and call it the 'adaptive target attack.' We provide a detailed theoretical analysis of the attack efficiency in the general RL setting and conduct complementary experiments in the DRL setting. The empirical results verify that our attack can successfully mislead the agent to a target policy with limited budgets and computational resources. Our attack works for general RL problems with both discrete and continuous action spaces and applies to diverse learning algorithms. We experimentally verify the efficiency of our attack on various popular DRL environments for continuous and classical control problems, including HalfCheetah, Hopper, Walker, and Acrobot, and various state-of-the-art DRL algorithms, including SAC (Haarnoja et al., 2018), TD3 (Dankwa & Zheng, 2019), DDPG (Lillicrap et al., 2015), and DQN (van Hasselt et al., 2016). We consider various target policies to show that our attack works in a wide range of practical scenarios. Among all the attack settings we consider, our results show that our black-box attack can always successfully make an agent take actions close to target actions during training with very limited budgets.

2 Background

Online reinforcement learning: We consider a standard and general online reinforcement learning setting consisting of an environment and a learning agent (Sutton & Barto, 2018). An environment is characterized by an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mu_0)$. Here $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(S)$ is the state transition function where $\Delta(S)$ is the space of state distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and μ_0 is the distribution of the initial states. Formally, the training process for the agent consists of multiple episodes. In each episode, the environment is initialized at a state s sampled from $s \sim \mu_0$. The agent takes an action a, and the environment evolves to the next state $s' \sim \mathcal{P}(s, a)$ and returns an instant reward signal $r \sim \mathcal{R}(s, a)$. At each time step, if there is no poisoning attack, the agent will receive the actual observation tuple (s, a, s', r). A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ for the environment is a mapping from a state to a probability distribution over the action space. Let μ^π be the state distribution in an episode where the actions are always sampled from the policy π at the states for each time step. The performance of a policy π is then defined as $\mathcal{J}_{\mathcal{R}}(\pi) := \mathbb{E}_{s\sim\mu^\pi, a\sim\pi(s)}\mathcal{R}(s, a)$. The optimal policy is the policy with the highest performance. Since a deterministic optimal policy always exists (Sutton & Barto, 2018), we use π^* to denote the deterministic optimal policy and $\pi^*(s)$ to represent the optimal action at a state s.

Threat model: We consider an adversary that can inject perturbations into the reward during agent training. At each time step t, the current state is s^t, and the action taken by the agent is a^t. The true feedback of the environment is next state s^{t+1} and reward r^t. Before the agent observes the true feedback (s^t, a^t, s^{t+1}, r^t), the adversary observes it and injects perturbation Δ^t to the reward. Then, the agent receives $(s^t, a^t, s^{t+1}, r^t + \Delta^t)$. We denote $\mathbf{\Delta} = \{\Delta^1, \dots, \Delta^T\}$ as the attacker's perturbation strategy. It is a random variable depending on the observations of the interaction between the agent and the environment.

Constraints on the attack: We want the attack to work in a black box setting with minimal information about the learner and environment. We formally describe the constraints of a realistic attack below.

1. **Oblivious to the learning agent & Universal attack**: The attacker is oblivious to the learning algorithms and setups of the agent. The attack should universally work on different algorithms the agents use.
2. **Oblivious to the environment**: The attacker has no knowledge of the transition function $\mathcal{P}$ and reward function $\mathcal{R}$ of the environment. It is aware of the state space $\mathcal{S}$ and action space $\mathcal{A}$.

Budgets required by the attack: The efficiency of the attack depends on the budgets required by the attacker. In this work, we consider the following attack budgets.

1. The total amount of the reward perturbation $||\mathbf{\Delta}||_1 = \sum_{t=1}^{T} |\Delta^t|$ during the training process the attacker applies.

2. The maximal amount of reward perturbation $||\mathbf{\Delta}||_\infty = \arg\max_{t\in[T]} |\Delta^t|$ at each step. This ensures the perturbation at each step is bounded.

The goal of the attack: In general, the attacker wants the agent to follow certain target behaviors during training that are characterized by a deterministic target policy $\pi^\dagger$ (Sun et al., 2020; Zhang et al., 2020). The action given by the target policy at a state is the target behavior that the attacker wants the agent to take at the state. Denote a^t as the action taken by the agent under the attack and $\pi^\dagger(s^t)$ as the target action. Let $d(\cdot,\cdot) : \mathcal{A} \times \mathcal{A} \to [0, 1]$ be a distance measure between two actions. Taking the continuous action space as an example, the distance can be defined based on L2 distance as $d(a_1, a_2) = \|a_1 - a_2\|_2/L$, where $L = \max_{x,y\in\mathcal{A}} \|x - y\|_2$ is the maximum 2-norm distance between any two actions. We adopt this definition for the experiments in Section 4. Note that the analysis and the method in this work apply to general distance measures. Denote $d^t = d(a^t, \pi^\dagger(s^t))$ as the distance between the agent's action and the target action at timestep t and $\mathbf{d} = \{d^1, \dots, d^T\}$ as the distances during the training process. Formally, the goal of the attack is set as follows:

The attack aims to induce a minimal difference between the actions selected by the agent and the target actions during training $||\mathbf{d}||_p = \left[\sum_{t=1}^{T}(d^t)^p\right]^{1/p}$ with respect to a p-norm.

We want to find an attack that works under the constraints and achieves the goal with limited budgets. In the next section, we formulate the attacker's goal as an optimization problem and develop efficient attack methods.

3 Adaptive Target Attack and Analysis

3.1 Reward poisoning attack as an optimization problem

Let T be the total number of training steps, $\mathcal{L}$ be the class of learning algorithms from which the agent will choose its learning algorithm. Denote the state and selected action at time t during training under the attack as s^t and a^t. The perturbation at time t is denoted as Δ^t, which is a random variable determined by the attack strategy depending on the previous observation $\{s^{1:t}, a^{1:t}, r^{1:t-1}, \Delta^{1:t-1}\}$ during training. The attack strategy is represented by the perturbations $\mathbf{\Delta} = \{\Delta^1, \dots, \Delta^T\}$.

We focus on universal black box attacks. The attack should be efficient against any algorithm Alg $\in \mathcal{L}$ that the agent might use without knowing exactly which algorithm is used. Ideally, an efficient attack should minimize: 1. the distance ϵ between actions taken by the agent with any Alg $\in \mathcal{L}$ and the target actions 2. the total perturbation C, and 3. the maximum perturbation at each step B. However, such an optimal attack might not exist as an attack could induce a smaller value of ϵ with more budget on B and C. Therefore, we define the most efficient attack for a fixed perturbation budget B, C in Eq 1.

$$\begin{aligned}
&\min_{\Delta^{t=1,\dots,T}} \epsilon, \text{ s.t. } \forall \text{Alg} \in \mathcal{L}, \\
&\mathbb{E}_{\text{Alg},\mathcal{M}}[\|\mathbf{d}\|_p] \le \epsilon, \\
&\mathbb{E}_{\text{Alg},\mathcal{M}}[\|\mathbf{\Delta}\|_1] \le C, \mathbb{E}_{\text{Alg},\mathcal{M}}[\|\mathbf{\Delta}\|_\infty] \le B
\end{aligned} \tag{1}$$

To complete the definition of the attack problem, we need to define $\mathcal{L}$, the class of algorithms the agent might use. First we argue that letting $\mathcal{L}$ include all possible algorithms makes the problem trivial and is meaningless for the attacker. In this case, there always exists an algorithm that takes specific actions regardless of the environment and the attack to maximize ϵ. Such an algorithm dominates the value of ϵ and makes it the same for all attacks. As a result, any attack that satisfies the budget constraint will be an optimal solution, which is meaningless. It is also pointless for the attacker to consider attacking an inefficient and not data-driven algorithm that any rational agent will not use. Therefore, it is more meaningful for the attacker to consider agents using 'efficient' algorithms. To quantify which kind of algorithms are 'efficient,' we generalize a common feature shared by existing efficient DRL methods in the definition below.

Definition 3.1. (δ_p-efficient learning algorithm) Let the total training steps be T and the optimal policy of the environment be π^*. A δ_p-efficient learning algorithm guarantees that, in expectation, it will take the actions close to the optimal actions for most of the time during training, that is, $\mathbb{E}[\|\mathbf{d}\|_p] \leq \delta_p$, where $\delta_p > 0$ is a constant, and $\mathbf{d} = \{d^1, \ldots, d^t\}$ is the distance between the agent's action and the target action at each time step as previously defined.

In general, an efficient learning algorithm should be capable of identifying a majority of optimal actions or actions close to them within T steps in expectation. During training, it will take actions that are close to the optimal actions most of the time. We argue that existing efficient DRL algorithms are likely to be δ_p-efficient with a small value of δ_p. First, the goal of designing learning algorithms is to guarantee that the learning algorithm can find the optimal actions for a large set of environments. Second, a reinforcement learning algorithm needs to balance the exploration-exploitation trade-off, making it necessary to explore the sub-optimal actions strategically. In practice, most algorithms select the empirically optimal action with the highest probability, such as ϵ-greedy exploration (van Hasselt et al., 2016). So, the agent will identify the optimal actions or actions close to them as empirically best actions and correspondingly select those actions for most of the timesteps. This is indirectly supported by our empirical observations in Section 4. For the rest of this section, we consider $\mathcal{L}$ to be the set of δ_p-efficient learning algorithms in Eq 1.

Given the optimization problem in Eq 1, it is natural to quantify the efficiency of an attack as below:

Definition 3.2. ((ϵ, C, B) efficient attack) An attack is (ϵ, C, B) efficient if its perturbation $\mathbf{\Delta}$ satisfies the following conditions:

$$\begin{aligned}
&\forall \text{Alg} \in \mathcal{L}, \\
&\mathbb{E}_{\text{Alg},\mathcal{M}}[\|\mathbf{d}\|_p] \leq \epsilon, \\
&\mathbb{E}_{\text{Alg},\mathcal{M}}[\|\mathbf{\Delta}\|_1] \leq C, \mathbb{E}_{\text{Alg},\mathcal{M}}[\max_t]\|\mathbf{\Delta}\|_\infty] \leq B
\end{aligned} \tag{2}$$

An attack is more efficient if it is (ϵ, C, B)-efficient with a lower value of ϵ for the same value of B and C. By Eq 1, finding the optimal attack is equivalent to finding an (ϵ, C, B)-efficient attack with the minimal value of ϵ given B and C. Next, we show that finding the efficiency of an arbitrary attack strategy is hard.

First, the attacker needs to predict the influence of the attack on all δ_p-efficient algorithms from $\mathcal{L}$. There can be infinitely many δ_p-efficient algorithms, and the attacker does not know exactly what they are. Second, predicting the actions taken by a specific unknown learning algorithm Alg is still hard. The attacker needs to know the learning algorithm and the environment dynamics to predict the actions the agent takes and the state transitions in the future. However, it can only observe the state transitions and reward signals during training in the black box setting. It is hard to approximate the environment accurately with such limited observations. With no further limitation on the learning algorithm, the learning strategy at different timesteps can be independent, making it impossible to predict the agent's future strategy through its behavior in history.

For a targeted attack that aims at misleading the agent to follow a specific behavior, it is unreliable to adopt an attack strategy if the agent's behavior under the attack is unpredictable. Therefore, it is crucial to identify the attacks that predictably influence the learning agent's behavior. Next, we show a class of attacks that satisfy this property and find an efficient instance from the class with minimal per-step corruption.

3.2 Efficient adversarial MDP attack framework

Here, we build an attack framework called 'efficient adversarial MDP attack' so that the efficiency of any attack instances in the framework can be estimated. We formally define the framework in Definition 3.3.

Definition 3.3. (Efficient Adversarial MDP attack) An adversarial MDP attack specifies an adversarial environment $\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \widehat{\mathcal{R}})$, which has the same state space, action space, and state transition function as the true environment but a different reward function $\widehat{\mathcal{R}}$. The perturbation at time t is $\Delta^t = \widehat{\mathcal{R}}(s^t, a^t) - \mathcal{R}(s^t, a^t)$. Under the attack, the agent will receive reward $\widehat{\mathcal{R}}(s^t, a^t)$ at time t. Therefore, the agent is actually trained in the adversarial $\widehat{\mathcal{M}}$ under the attack. In addition, an efficient adversarial MDP attack characterized by $\widehat{\mathcal{M}}$ should satisfy the conditions below for a perturbation bound $\Delta > 0$.

1. The target policy $\pi^\dagger$ is the optimal policy under the adversarial environment $\widehat{\mathcal{M}}$.

2. The differences between the adversarial and true reward functions are small for the actions whose distance to the target actions is small at any state. Given a value of $\Delta \in \mathbb{R}^+$, for all state $s \in \mathcal{S}$, $|\widehat{\mathcal{R}}(s,a) - \mathcal{R}(s,a)| \leq \Delta \cdot d(a, \pi^\dagger(s))^p$.

We denote EM as the class of all efficient adversarial MDP attacks. We denote an instance of EM as M(Δ), where Δ is its perturbation bound. In Theorem 3.4, we show a guarantee on the attack efficiency of an efficient adversarial MDP attack M(Δ) $\in$ EM.

Theorem 3.4. *For an efficient adversarial MDP attack M(Δ) $\in$ EM, it satisfies equation* (1) *with $\epsilon \leq \delta_p$, $B = \Delta$, and $C \leq (\delta_p)^p \cdot \Delta$.*

The proof for Theorem 3.4 can be found in Appendix A. Here, we explain how the framework is constructed and how its efficiency is evaluated. The framework is built to work closely with the δ_p-efficient learning assumption. We note that the assumption describes the behavior of the agent learning from a static environment. Therefore, this assumption applies if the attacker makes the agent train as if in a malicious environment $\widehat{\mathcal{M}}$ obtained by perturbing the true environment $\mathcal{M}$. In this case, for any algorithm Alg $\in \mathcal{L}$, it is always true that $\mathbb{E}[\sum_{t=1}^{T} \big(d(a^t, \hat{\pi}^*(s^t))^p\big)^{1/p}] \leq \delta_p$, where $\hat{\pi}^*$ is the adversarial optimal policy on the adversarial environment. If the adversarial optimal policy is exactly the target policy $\hat{\pi}^* = \pi^\dagger$, then the attack guarantees that $\epsilon = \max_{\mathrm{Alg} \in \mathcal{L}} \mathbb{E}_{\mathrm{Alg}, \mathcal{M}, \Delta^{t=1,\dots,T}}[\|\mathbf{d}\|_p] \leq (\delta_p)^p$. To make the agent train as if in $\widehat{\mathcal{M}}$, the perturbation at time T should satisfy $\Delta^t = \widehat{\mathcal{R}}(s^t, a^t) - \mathcal{R}(s^t, a^t)$. To limit the requirement budget, the attacker can limit the perturbation around the adversarial optimal policy, as the agent will mostly take those actions. The idea we adopt is to connect the perturbation applied to an action to its distance from the target action. More specifically, we require $|\widehat{\mathcal{R}}(s,a) - \mathcal{R}(s,a)| \leq \Delta \cdot d(a, \pi^\dagger(s))^p$. In this case, for any algorithm Alg $\in \mathcal{L}$, we have $B = \Delta$ directly by the attack definition. The total perturbation C applied by the attack is connected to the total distance between the agent's actions and target actions: $\sum_t |\widehat{\mathcal{R}}(s^t, a^t) - \mathcal{R}(s^t, a^t)| \leq \Delta \cdot \|\mathbf{d}\|_p \leq (\delta_p)^p \cdot \Delta$.

The guarantees above suggest that an attack could be more efficient with a smaller value of Δ. However, the value of Δ must be significant enough to make the target policy $\pi^\dagger$ optimal in the adversarial environment $\widehat{\mathcal{R}}$. Next, we aim to find this threshold and the corresponding efficient adversarial MDP attack with the minimal value of Δ.

3.3 Efficient adversarial MDP attack with minimal Δ: adaptive target attack

This section shows an instance of the efficient adversarial MDP attack framework that requires the minimal value of Δ. First, we define the 'adaptive targeted attack' in Definition 3.5 below.

Definition 3.5. (Adaptive target attack) The adaptive target attack AT(Δ, q) as described in Algorithm 1 is parameterized by the maximal reward perturbation $\Delta \in \mathbb{R}^+$, and a positive value $q \in \mathbb{R}^+$. Given a target policy $\pi^\dagger$ and a distance measure d, the reward perturbation at round t by the adaptive target attack satisfies $\Delta^t = -\Delta \cdot d(a, \pi^\dagger(s))^q$.

The adaptive target attack penalizes the agent for taking an action different from the target action. More specifically, the penalty Δ^t for an action a^t is a polynomial function of its distance to the target action: $\Delta^t = \Delta \cdot d(a^t, \pi^\dagger(s^t))^q$. The computational resource required by the attack is the memory to store the target policy $\pi^\dagger$, and the computational power to calculate the target action given by $\pi^\dagger$ and its distance to the action taken by the learning agent.

Formally, in Theorem 3.6, we show that the efficient adversarial MDP attacks with the minimal value of perturbation bound Δ^* must be an adaptive target attack with $\Delta = \Delta^*$ and $q = p$. The proof can be found in the Appendix A. Intuitively, in the adversarial environment, the adaptive target attack with $q = p$ penalizes all non-targeted policies more than any other instances of efficient adversarial MDP attack with the same value of Δ, making it easier to highlight the target policy.

Theorem 3.6. *Let EM be the class of efficient adversarial MDP attacks and M(Δ) be an instance with perturbation bound Δ. Denote $\Delta^* = \min_{M(\Delta) \in EM} \Delta$ as the minimal value of Δ for an efficient adversarial MDP attack. The value of Δ^* is*

$$\Delta^* = \max_{\pi : D(\pi, \pi^\dagger) > 0} \frac{\mathcal{J}_{\mathcal{R}}(\pi) - \mathcal{J}_{\mathcal{R}}(\pi^\dagger)}{D(\pi^\dagger, \pi)},$$

Algorithm 1 Adaptive Target Attack Framework

```
**Input**: target policy $\pi^\dagger$
**Params**: distance measure $d$, maximal per-step perturbation $\Delta$, polynomial factor $q$
**for** $t = 1, 2, \dots T$ **do**
    Observe environment state $s^t$, agent's action $a^t$, and reward signal $r^t$
    Perturb reward signal $r^t \leftarrow r^t - \Delta \cdot d(a^t, \pi^\dagger(s^t))^q$
**end for**
```

where $D(\pi_1, \pi_2) := \mathbb{E}_{s \sim \mu^{\pi_1}} d(\pi_1(s), \pi_2(s))$ *represents the distance between two polices. In addition, the adaptive target attack* $AT(\Delta^*, p)$ *is the adversarial MDP attack with minimal perturbation bound* $M(\Delta^*)$*.*

In summary, Theorem 3.4 guarantees a lower bound on the attack efficiency of an adversarial MDP attack in Definition 3.3. The lower bound decreases as the perturbation bound Δ of an adversarial MDP attack decreases, suggesting that efficient adversarial MDP attacks should have low values of Δ. We further find that the adversarial MDP attack with the minimal value of Δ is an adaptive target attack in Definition 3.5 with $q = p$. Therefore, we believe an adaptive target attack is efficient with a sufficient value of Δ and $q = p$. However, note that Theorem 3.4 provides a bound on the attack efficiency instead of the exact value. So, an adversarial MDP attack with a lower value of Δ does not necessarily be less efficient in practice. Also, under the (ϵ, B, C) definition for attack efficiency, the efficiencies of any two attacks are not directly comparable as there can be a trade-off between ϵ and the budgets B, C. Therefore, the versions of adaptive target attacks with $q \neq p$ might also be efficient in practice, and we empirically investigate this in the next section.

4 Experiments

Although the assumption that the algorithm used by the agent is an efficient learning algorithm in Definition 3.1 may not strictly hold in practice, we demonstrate in this section that our adaptive target attack is effective in the universal learning scenarios we test. In the theoretical analysis in Section 3, we study the general case where the attacker aims to minimize the total distances $\|\mathbf{d}\|_p$ between the agent's actions and the target actions under an arbitrary p-norm. For the simplicity of empirical evaluation, we focus on a natural case where $p = 1$, that is, the linear summation of the distances at each timestep $\|\mathbf{d}\|_1 = \sum_{t=1}^{T} d(a^t, \pi^\dagger(s^t))$ where a^t is agent's action and $\pi^\dagger(s^t)$ is the target action. Accordingly, we set $q = 1$ for the adaptive target attack in most of our experiments, and we study the attack with different values of q in the ablation study. Note that our attack applies to general RL settings. The rest of this section mainly focuses on the standard continuous control problems with continuous action spaces. In Appendix B, we show the attack setup and the empirical evaluations in the RL environments with discrete action spaces.

4.1 Experiment Setup

Learning scenarios and target policies: In this work, we focus on the standard continuous robotic control problems from Mujoco (Todorov et al., 2012), including HalfCheetah, Hopper, and Walker. We verify that our attack works efficiently against the environments with discrete action spaces in Apppendix B. For the learning algorithms, we consider state-of-the-art DRL algorithms, including DDPG (Lillicrap et al., 2015), TD3 (Dankwa & Zheng, 2019), and SAC (Haarnoja et al., 2018). The number of total training steps is set as 6×10^5 to ensure that the algorithms can collect high rewards on average during training when there is no attack.

Our attack is designed to work for universal target policies. Therefore, we adopt the idea of the random, medium, and expert policies from (Fu et al., 2020) to represent policies with diverse performances. A random policy is randomly generated and is usually associated with low performance. An expert policy is a policy of the highest performance that an efficient learning algorithm can learn. A medium policy has a performance in the middle of the above two. We construct such policies following the process in (Fu et al., 2020). We mainly focus on the medium policy in most of the experiments and test all three types of policies in the ablation study. We believe that setting the target policy as a medium policy is relatively more realistic. A target policy usually has some behavior desired by the attacker but not desired by the agent, so its performance should be sub-optimal. However, if the performance of a target policy is comparable to a randomly generated policy, then the agent will find no improvement during training and may discard the trained policy in the end. Therefore, a medium policy is relatively more suitable to be the target. In Appendix C, we show the details of the behavior of the medium policy we choose as a target for the HalfCheetah environment. In general, we show that the robot controlled by the medium policy exhibits an unnatural behavior compared to that controlled by an expert policy. We further show that the agent learns the same strange behavior under the adaptive target attack with the medium policy as the target.

Baseline Attacks: To the best of our knowledge, there is no existing baseline for black-box targeted online DRL attack. We adopt a state-of-the-art black-box online DRL attack from Xu et al. (2023) to our setting as a baseline, which we call the 'neighborhood attack.' The neighborhood attack is parameterized by $\Delta \in \mathbb{R}^+$ and $r \in \mathbb{R}^+$, and its corruption strategy is $\Delta^t = \Delta \cdot \mathbb{1}\{d(a^t, \pi^\dagger(s^t)) \geq r\}$. That is, it applies a fixed penalty when the agent takes an action far from the target action and no penalty otherwise. In addition, we consider a random attack strategy that randomly decides corruption at each training step as a naive baseline. More specifically, the random attack is parameterized by $\Delta \in \mathbb{R}^+$. At each training step, the corruption it applies is uniformly at random drawn from $[-\Delta, \Delta]$. In Appendix E, we provide an additional comparison to the white-box online DRL attack 'va2c-p' from Sun et al. (2020). We discuss the difference between their attacks and ours in Section 5

Attack Scenarios: We consider two meaningful scenarios for evaluating the performance of an attack. In the first scenario, there is no hard limit on the attacker's budget, so the attack can always follow its strategy to apply perturbation. In this case, we can measure the attack's efficiency (ϵ, C, B) according to Equation 2 to verify our theoretical analysis in Section 3. In the second scenario, we force a hard limit on the attacker's budget so that an attacker can never require a budget C more than a specific value. In this scenario, all attacks will have the same constraints on the attack budget B and C, making it more straightforward to compare the performance between attacks through the difference in ϵ.

4.2 Efficiency of Attacks

First, we investigate the efficiency of attacks in different learning scenarios. The main goal is to verify that our adaptive target attack is universally effective against different efficient learning algorithms. For this purpose, we consider three environments: HalfCheetah (HC), Hopper (HP), and Walker (WK), learned by the three algorithms: DDPG, TD3, and SAC. Here, we only consider the target policy to be a medium policy, as previously discussed. In the ablation study, we test the target policy of other types.

Scenario 1: Attacks with no hard limit on budgets In this scenario, the attacker can always follow its corruption strategy. To ensure a fair comparison, all the attacks share the same value of maximal per-step corruption. For the HalfCheetah environment, the maximal per-step corruption is set as $B = 50$; for the Hopper and Walker environments, the value is set as $B = 20$. Correspondingly, all the attacks set $\Delta = B$ to utilize the per-step corruption budget fully. In the ablation study, we test the attack with smaller values of Δ. For the baseline neighborhood attack, we test with $r = 1, 1.5, 2$ and show the best result. Each experiment is repeated for 10 times, and we report the average together with the confidence intervals.

The results are shown in Table 1. For clarity, we show the average distance ϵ/T and corruption C/T over each time step. For our adaptive target attack, in all learning scenarios, it can always make the agent closely follow the target behavior (low value of ϵ) with a limited budget required (low value of C). Note that the variance of learning results under the attack is also small, implying that the influence of the attack on the agent is stable. This indirectly verifies our assumption of the efficient learning algorithms and efficiency guarantees on our attack. More importantly, the attack results are consistent for all the three learning algorithms we test with, suggesting that our attack is universally efficient against different efficient learning algorithms. In comparison, for both baseline attacks, they can hardly make the agent follow the target behavior compared to the case with no attack while spending a lot of budget.

Scenario 2: Attacks with a hard limit on budgets In this scenario, the attack has to stop applying perturbation after the total perturbation reaches C. For the HalfCheetah environment, we set $C/T = 5$; for the Walker and Hopper environments, we set $C/T = 3$. This setup ensures that the attacks we consider will

Env-Alg	Clean ($\frac{\epsilon}{T}$)	**Adapt** ($\frac{\epsilon}{T}$)	Neigh ($\frac{\epsilon}{T}$)	Random ($\frac{\epsilon}{T}$)	**Adapt** ($\frac{C}{T}$)	Neigh ($\frac{C}{T}$)	Random ($\frac{C}{T}$)
HC-DDPG	0.52(0.05)	**0.09**(**0.01**)	0.46(0.08)	0.62(0.01)	**4.58**(**0.41**)	49.99(0.01)	25.00(0.01)
HC-TD3	0.51(0.07)	**0.10**(**0.02**)	0.49(0.04)	0.54(0.01)	**5.24**(**0.49**)	49.99(0.01)	24.99(0.01)
HC-SAC	0.52(0.02)	**0.12**(**0.01**)	0.46(0.05)	0.52(0.00)	**5.75**(**0.76**)	49.99(0.00)	25.02(0.02)
HP-DDPG	0.51(0.01)	**0.17**(**0.00**)	0.61(0.01)	0.52(0.00)	**3.34**(**0.02**)	20.00(0.00)	9.99(0.01)
HP-TD3	0.45(0.03)	**0.17**(**0.01**)	0.61(0.02)	0.57(0.02)	**3.37**(**0.05**)	20.00(0.00)	10.00(0.01)
HP-SAC	0.43(0.01)	**0.16**(**0.00**)	0.56(0.01)	0.46(0.01)	**3.30**(**0.02**)	20.00(0.00)	10.00(0.01)
WK-DDPG	0.49(0.02)	**0.16**(**0.01**)	0.58(0.01)	0.51(0.04)	**3.13**(**0.14**)	20.00(0.00)	10.01(0.01)
WK-TD3	0.54(0.07)	**0.17**(**0.00**)	0.58(0.00)	0.46(0.06)	**3.38**(**0.06**)	20.00(0.00)	10.00(0.01)
WK-SAC	0.41(0.05)	**0.17**(**0.02**)	0.47(0.00)	0.39(0.03)	**3.46**(**0.31**)	20.00(0.00)	10.00(0.01)

Table 1: Performance of attacks in different learning scenarios without a limit on budget C. In the 'Dataset-Alg' column, 'HC' represents HalfCheetah, 'HP' represents Hopper, and 'WK' represents Walker. For other columns, 'Adapt' represents the adaptive target attack, 'Neigh' represents the neighborhood attack, and 'Random' represents the random attack. The values in the brackets are the variance.

likely run out of budget during the training process. In the ablation study, we test the attacks with different budgets of C. The results are shown in Table 2. We find that both baseline attacks cannot mislead the agent better than no attack, suggesting that this attack scenario is challenging. Recall our analysis in Section 3 states that with a sufficient value of B, the adaptive attack can achieve small values of ϵ and only requires a small budget of C. Here, our empirical results further show that even when the attack stops applying corruption in the middle of training, our adaptive attack can still make the agent mostly select target actions. The reason is that our attack already makes the agent learn most of the target actions as the optimal actions before running out of budget. So when there is no more perturbation, the agent will only seldom take actions far from the target actions, and it will take a long time for the agent to gather enough unperturbed new information to reverse the impact of previous corrupted information on the actions far from target actions.

Dataset-Alg	Clean ($\frac{\epsilon}{T}$)	**Adapt** ($\frac{\epsilon}{T}$)	Neigh ($\frac{\epsilon}{T}$)	Random ($\frac{\epsilon}{T}$)
HC-DDPG	0.52(0.05)	**0.09**(**0.00**)	0.59(0.03)	0.62(0.01)
HC-TD3	0.51(0.07)	**0.11**(**0.02**)	0.62(0.09)	0.54(0.01)
HC-SAC	0.52(0.02)	**0.13**(**0.04**)	0.56(0.04)	0.52(0.02)
HP-DDPG	0.51(0.01)	**0.19**(**0.00**)	0.64(0.07)	0.54(0.04)
HP-TD3	0.45(0.03)	**0.18**(**0.00**)	0.62(0.07)	0.47(0.01)
HP-SAC	0.43(0.01)	**0.17**(**0.00**)	0.58(0.04)	0.45(0.01)
WK-DDPG	0.49(0.02)	**0.15**(**0.01**)	0.55(0.03)	0.52(0.05)
WK-TD3	0.54(0.07)	**0.18**(**0.01**)	0.59(0.03)	0.46(0.04)
WK-SAC	0.41(0.05)	**0.20**(**0.00**)	0.44(0.01)	0.42(0.04)

Table 2: Performance of attacks in different learning scenarios with a hard limit on budget C.

4.3 Ablation Study

In this section, we focus on measuring the efficiency of the attacks according to Definition 3.2 in different setups of the attack and the learning scenario. In Appendix B, we provide additional ablation studies on total corruption C and per-step perturbation Δ and also test the attack in the RL environments with discrete action spaces.

Different types of target policies: Here, we verify that the adaptive target attacks remain efficient for the dataset collected by different types of policies. We take the HalfCheetah environment as an example and test the attack with target policies being random, medium, and expert policies. We set $\Delta = 50$ and $C/T = 5$ as in the main experiments. We measure the efficiency of the attack and show the results in Table 3. We observe that the adaptive target attack always successfully misleads the agent to follow the target behavior of all kinds of target policies. The results also show that the difficulties of misleading the agent to different target behaviors differ. The random policy is the easiest target as the attacker can make the agent take actions very close to the target actions (low value of ϵ) while using the lowest corruption budget of C among all targets. In comparison, the expert policy is the most challenging.

Alg-Target	Clean ($\frac{\epsilon}{T}$)	Adapt ($\frac{\epsilon}{T}$)	Adapt ($\frac{C}{T}$)
DDPG-Rand	0.51(0.01)	0.06(0.00)	2.76(0.01)
TD3-Rand	0.54(0.02)	0.06(0.00)	2.80(0.02)
SAC-Rand	0.43(0.01)	0.03(0.00)	1.66(0.02)
DDPG-Med	0.52(0.05)	0.09(0.01)	4.58(0.41)
TD3-Med	0.51(0.07)	0.10(0.02)	5.24(0.49)
SAC-Med	0.52(0.02)	0.12(0.01)	5.75(0.76)
DDPG-Exp	0.52(0.07)	0.15(0.02)	7.36(1.07)
TD3-Exp	0.45(0.09)	0.17(0.01)	8.45(0.66)
SAC-Exp	0.42(0.05)	0.14(0.04)	6.98(1.90)

Table 3: Performance of the adaptive target attack for different types of target policies when there is no hard limit on C.

Different values of q: Here, we study the efficiency of adaptive target attack when setting q to be different values. We test the attack with $q = \{0.5, 1, 2, 4\}$, while the other setups are the same as the experiments in Section 4.2. The results are shown in Table 4. We observe a trade-off that as the value of q increases, the required budget on C decreases, but the influence on the learning agent ϵ increases. In the case we test with, when the value of q is no greater than 1, decreasing the value of q has a marginal influence on ϵ, but the increment on C is significant, which is undesired by the attacker. Similarly, when the value of q is no less than 2, increasing the value of q has limited influence on C, but the increment on ϵ is significant. Therefore, $q \in [1, 2]$ intuitively balances the trade-off better than other choices.

Alg-p	Clean ($\frac{\epsilon}{T}$)	Adapt ($\frac{\epsilon}{T}$)	Adapt ($\frac{C}{T}$)
DDPG-0.5	0.52(0.05)	0.10(0.01)	13.59(0.76)
TD3-0.5	0.51(0.07)	0.12(0.01)	15.01(0.41)
SAC-0.5	0.52(0.02)	0.12(0.01)	14.39(0.93)
DDPG-1	0.52(0.05)	0.09(0.01)	4.58(0.41)
TD3-1	0.51(0.07)	0.10(0.02)	5.24(0.49)
SAC-1	0.52(0.02)	0.12(0.01)	5.75(0.76)
DDPG-2	0.52(0.05)	0.11(0.01)	1.29(0.12)
TD3-2	0.51(0.07)	0.13(0.02)	1.22(0.11)
SAC-2	0.52(0.02)	0.12(0.01)	1.45(0.30)
DDPG-4	0.52(0.05)	0.21(0.02)	0.68(0.13)
TD3-4	0.51(0.07)	0.19(0.01)	0.49(0.03)
SAC-4	0.52(0.02)	0.21(0.09)	0.59(0.02)

Table 4: Performance of the adaptive target attack for different p-norm when there is no hard limit on C.

In this work, we do not focus on the agent's ability to detect abnormal data, which can be used to defend against data corruption. To the best of our knowledge, it remains unknown how to detect abnormal data induced by a poisoning attack during training time in an online RL setting. In Appendix D, we discuss anomaly detection at training time in RL in detail.

5 Related Work

Data poisoning attack in simpler reinforcement learning Data poisoning attacks in reinforcement learning have been considered in simpler bandit cases (Jun et al., 2018; Garcelon et al., 2020; Liu & Shroff, 2019) and tabular MDP cases (Rakhsha et al., 2020; Xu et al., 2021; Zhang et al., 2020; Banihashem et al., 2022; Liu & Lai, 2021; Rakhsha et al., 2021; Li et al., 2024). Since the deep MDP case has a much more complicated environment to learn from, the attacks mentioned here cannot be applied to the deep MDP case. Banihashem et al. (2022); Rakhsha et al. (2020) adopt the same basic attack framework as our work. Besides the difference in the definition of the attack's budget, the main limitation in their works is that their methods are only defined on discrete state and action spaces and require full knowledge of the environment. The attack problems proposed in the works are also impossible to solve in the deep MDP case due to the complexity of the environment.

Observation perturbation attack in deep reinforcement learning A line of work study observation perturbation attack and defense in DRL (Behzadan & Munir, 2017a;b; Inkawhich et al., 2019; Liang et al., 2022; Liu et al., 2022; Qiaoben et al., 2024). There are three main differences between these works and ours. First, observation perturbation attacks perturb the state signal, while our attack perturbs the reward signal. Second, observation perturbation attacks do not change the environment's dynamics but change the environment's signal observed by the agent. Our attack changes the dynamics of the environment. Third, the observation perturbation attack focuses on the intrinsic generalization vulnerabilities of the neural networks used by the learning agent. In contrast, our attack focuses on the vulnerabilities of the exploration strategy of the learning agent.

Data poisoning attack in deep reinforcement learning Xu et al. (2023) study untargeted reward poisoning attacks against DRL. Sun et al. (2020) is the only other work considering targeted attacks in DRL. There are three main limitations of this attack compared to ours and therefore less practical: (a) the attack is white-box that requires the knowledge of the learning algorithm, (b) the attack only works for on-policy learning algorithms, and (c) the attacker decides the perturbation after a batch of training steps. In addition, their attack problem is designed to minimize the cost of corruption per training batch rather than each training step. Our attack is designed to minimize the per-step corruption and total corruption, making our attack more general. For empirical evaluation, they only cover naive target policies that are not representative of practical attack scenarios. In our work, we experiment with multiple non-trivial types of target policies.

6 Conclusion and Limitations

In this work, we propose the first black-box targeted reward poisoning attack against online DRL. Our attack is practical because (1) it does not have knowledge of the learning agent and environment dynamics, (2) it requires a limited budget for corruption over each step and the whole training process, and (3) it requires limited computational resources. The limitations of our attack include: (1) it only studies reward poisoning attacks, (2) it only works for online reinforcement learning settings. We hope our study will motivate the development of DRL algorithms robust against data poisoning attacks in the future.

References

M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*, 2021.

Kiarash Banihashem, Adish Singla, Jiarui Gan, and Goran Radanovic. Admissible policy teaching through reward design. *arXiv preprint arXiv:2201.02185*, 2022.

Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In Petra Perner (ed.), *Machine Learning and Data Mining in Pattern Recognition - 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings*, volume 10358 of *Lecture Notes in Computer Science*, pp. 262–275. Springer, 2017a. doi: 10.1007/978-3-319-62416-7_19. URL `https://doi.org/10.1007/978-3-319-62416-7_19`.

Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017b.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Proc. Neural Information Processing Systems (NeurIPS)*, pp. 4299–4307, 2017.

Stephen Dankwa and Wenfeng Zheng. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, pp. 1–5, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Evrard Garcelon, Baptiste Roziere, Laurent Meunier, Jean Tarbouriech, Olivier Teytaud, Alessandro Lazaric, and Matteo Pirotta. Adversarial attacks on linear contextual bandits. *Advances in Neural Information Processing Systems*, 33:14362–14373, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Tom Haider, Karsten Roscher, Felippe Schmoeller da Roza, and Stephan Günnemann. Out-of-distribution detection for reinforcement learning agents with probabilistic dynamics models. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 851–859, 2023.

Matthew Inkawhich, Yiran Chen, and Hai Li. Snooping attacks on deep reinforcement learning. *arXiv preprint arXiv:1905.11832*, 2019.

Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Jerry Zhu. Adversarial attacks on stochastic bandits. *Advances in neural information processing systems*, 31, 2018.

Jianhui Li, Bokang Zhang, and Junfeng Wu. Online poisoning attack against reinforcement learning under black-box environments. *arXiv preprint arXiv:2412.00797*, 2024.

Yongyuan Liang, Yanchao Sun, Ruijie Zheng, and Furong Huang. Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning. *Advances in Neural Information Processing Systems*, 35:22547–22561, 2022.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Fang Liu and Ness Shroff. Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*, pp. 4042–4050. PMLR, 2019.

Guanlin Liu and Lifeng Lai. Provably efficient black-box action poisoning attacks against reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12400–12410, 2021.

Zuxin Liu, Zijian Guo, Zhepeng Cen, Huan Zhang, Jie Tan, Bo Li, and Ding Zhao. On the robustness of safe reinforcement learning under observational perturbations. *arXiv preprint arXiv:2205.14691*, 2022.

Robert Müller, Steffen Illium, Thomy Phan, Tom Haider, and Claudia Linnhoff-Popien. Towards anomaly detection in reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1799–1803, 2022.

Linas Nasvytis, Kai Sandbrink, Jakob Foerster, Tim Franzmeyer, and Christian Schroeder de Witt. Rethinking out-of-distribution detection for reinforcement learning: Advancing methods for evaluation and detection. *arXiv preprint arXiv:2404.07099*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

You Qiaoben, Chengyang Ying, Xinning Zhou, Hang Su, Jun Zhu, and Bo Zhang. Understanding adversarial attacks on observations in deep reinforcement learning. *Science China Information Sciences*, 67(5):152104, 2024.

Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. *CoRR*, abs/2003.12909, 2020. URL `https://arxiv.org/abs/2003.12909`.

Amin Rakhsha, Xuezhou Zhang, Xiaojin Zhu, and Adish Singla. Reward poisoning in reinforcement learning: Attacks against unknown learners in unknown environments. *arXiv preprint arXiv:2102.08492*, 2021.

Yanchao Sun, Da Huo, and Furong Huang. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. *arXiv preprint arXiv:2009.00774*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 2094–2100. AAAI Press, 2016. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389`.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Paweł Wawrzyński. A cat-like robot real-time learning to run. In *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*, pp. 380–390. Springer, 2009.

Hang Xu, Rundong Wang, Lev Raizman, and Zinovi Rabinovich. Transferable environment poisoning: Training-time attack on reinforcement learning. In *Proceedings of the 20th international conference on autonomous agents and multiagent systems*, pp. 1398–1406, 2021.

Yinglun Xu, Qi Zeng, and Gagandeep Singh. Efficient reward poisoning attacks on online deep reinforcement learning. *Trans. Mach. Learn. Res.*, 2023, 2023. URL `https://openreview.net/forum?id=25G63lDHV2`.

Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, pp. 11225–11234. PMLR, 2020.

A Proof

Proof for Theorem 3.4:

Proof. Since the target policy is also the optimal policy on the adversarial environment constructed by the attack $\hat{\pi}^* = \pi^\dagger$, directly by the definition of δ_p-efficient learning algorithms, we have $\mathbb{E}[\|\mathbf{d}\|_p/T] \leq \delta_p$.

Note that $d(\cdot,\cdot) \leq 1$ is always true, then by the definition of the attack, the perturbation at each step is at most $\Delta^t \leq \Delta \cdot d(a^t, \pi^\dagger(s^t)^p \leq \Delta$, suggesting that $B = \Delta$ for the second efficiency parameter of the attack.

The total amount of perturbation is bound by

$$\sum_{t=1}^{T} \Delta^T \leq \Delta \cdot \sum_{t=1}^{T} d(a^t, \pi^\dagger(s^t)^p = \Delta \cdot \|\mathbf{d}\|_p^p.$$

Therefore, we have

$$C \leq \mathbb{E}[\Delta \cdot \|\mathbf{d}\|_p^p] \leq \Delta \cdot (\delta_p)^p.$$

□

Proof for Theorem 3.6

Proof. Recall the performance of a policy π is $\mathcal{J}_{\mathcal{R}}(\pi) = \mathbb{E}_{s\sim\mu^\pi}\mathcal{R}(s,\pi(s))$. The performance of a policy π in the adversarial environment constructed by an efficient adversarial MDP attack $M(\Delta)$ can be decomposed as

$$\begin{aligned}
\mathcal{J}_{\widehat{\mathcal{R}}}(\pi) &= \mathbb{E}_{s\sim\mu^\pi}\widehat{\mathcal{R}}(s,\pi(s)) \\
&= \mathcal{J}_{\mathcal{R}}(\pi)\mathcal{R}(s,\pi(s)) - \mathbb{E}_{s\sim\mu^\pi}(\mathcal{R}(s,\pi(s)) - \widehat{\mathcal{R}}(s,\pi(s))) \\
&\geq \mathcal{J}_{\mathcal{R}}(\pi) - \Delta \cdot \mathbb{E}_{s\sim\mu^\pi} d(\pi(s), \pi^\dagger(s))^p \\
&:= \mathcal{J}_{\mathcal{R}}(\pi)(s) - \Delta \cdot D(\pi, \pi^\dagger)
\end{aligned} \tag{3}$$

Here we denote $D(\pi_1, \pi_2) = \mathbb{E}_{s\sim\mu^{\pi_1}} d(\pi_1(s), \pi_2(s))$. Note that if $D_s(\pi_1, \pi_2) = 0$, then π_1, π_2 always take the same actions at the states they can visit in an episode, which makes the two policies equivalent in effect. The inequality in the second last line can only take equality if the attack $M(\Delta)$ satisfies $\Delta^t = \Delta \cdot d(\pi(s^t), \pi^\dagger(s^t))^p$, which is also the case of adaptive target attack of $q = p$.

Based on the decomposition above, the difference between the performance in $\widehat{\mathcal{R}}$ of the target policy and any policy π with $D(\pi, \pi^\dagger) > 0$ at a state s can be written as

$$V_{\widehat{\mathcal{M}}}^{\pi^\dagger}(s) - V_{\widehat{\mathcal{M}}}^{\pi}(s) \leq (V_{\mathcal{M}}^{\pi^\dagger}(s) - V_{\mathcal{M}}^{\pi}(s)) + \Delta \cdot D_s(\pi, \pi^\dagger).$$

Therefore, the value of Δ to make the target policy has a higher performance than other policies $\pi : D(\pi^\dagger, \pi) > 0$ satisfies:

$$\Delta \geq \max_{\pi : D(\pi,\pi^\dagger)>0} \frac{(\mathcal{J}_{\mathcal{R}}(\pi) - \mathcal{J}_{\mathcal{R}}(\pi^\dagger)}{D(\pi^\dagger, \pi)} =: \Delta^*.$$

The inequality can take equality only when the adversarial MDP attack satisfies $\Delta^t = \Delta \cdot d(\pi(s^t), \pi^\dagger(s^t))^p$, which by definition is an adaptive target attack with the same value of $\Delta = \Delta^*$ and $q = p$. □

Alg-Δ	Clean ($\frac{\epsilon}{T}$)	Adapt ($\frac{\epsilon}{T}$)	Adapt ($\frac{C}{T}$)
DDPG-50	0.52(0.05)	0.09(0.01)	4.58(0.41)
TD3-50	0.51(0.07)	0.10(0.02)	5.24(0.49)
SAC-50	0.52(0.02)	0.12(0.01)	5.75(0.76)
DDPG-30	0.52(0.05)	0.09(0.01)	2.83(0.21)
TD3-30	0.51(0.07)	0.10(0.01)	3.01(0.26)
SAC-30	0.52(0.02)	0.11(0.01)	3.29(0.27)
DDPG-10	0.52(0.05)	0.16(0.03)	1.61(0.27)
TD3-10	0.51(0.07)	0.16(0.03)	1.61(0.29)
SAC-10	0.52(0.02)	0.16(0.02)	1.62(0.20)

Table 5: Performance of the adaptive target attack for different values of Δ without a hard limit on C.

B Additional Ablation Study

Different values of per-step corruption Δ

Here, we study the influence of the per-step corruption Δ on the adaptive target attack in both attack scenarios. We take the HalfCheetah environment and medium target policy as an example and test the attack with $\Delta = 10, 30, 50$. In the case with a hard limit on C, we set $C/T = 5$ as before.

In Table 5, we show the case where there is no hard limit on the total corruption C. With a higher value of Δ, the attacker can better mislead the agent to take actions close to the target actions, but it will also require the attacker to spend more corruption in total. Such a trade-off implies that increasing the value of Δ may not make the attack strictly more efficient. Especially in the case where there is a hard limit on C, increasing the value of Δ may not always make the attack more effective in misleading the agent. This is verified by our empirical evaluation in Table 6. When the value of Δ is too small, the attack is not effective. The reason is that the per-step corruption is too small to make the target policy optimal in the adversarial reward function. However, when Δ is big enough, increasing the value of Δ does not necessarily make the attack more efficient. Our empirical results show that the attacker can achieve a lower value of ϵ with $\Delta = 30$ compared to $\Delta = 50$. Based on the previous observation in the no hard limit on C scenario, the reason is that when Δ is large enough, the agent will gradually converge to always follow the target behavior. A higher value of Δ will not make the convergence process much faster but will require much more budget, making the attack stop applying corruption earlier.

Alg-Δ	Clean ($\frac{\epsilon}{T}$)	Adapt ($\frac{\epsilon}{T}$)
DDPG-50	0.52(0.05)	0.09(0.00)
TD3-50	0.51(0.07)	0.11(0.02)
SAC-50	0.52(0.02)	0.13(0.04)
DDPG-30	0.52(0.05)	0.09(0.00)
TD3-30	0.51(0.07)	0.09(0.01)
SAC-30	0.52(0.02)	0.10(0.00)
DDPG-10	0.52(0.05)	0.15(0.03)
TD3-10	0.51(0.07)	0.15(0.03)
SAC-10	0.52(0.02)	0.18(0.01)

Table 6: Performance of the adaptive target attack for different values of Δ with a hard limit on C.

Different values of hard limit on budget C

Here, we test the performance of the adaptive target attack given different values of hard limit on budget C. We take the HalfCheetah environment as an example, set the target policy to be the medium policy, and set $\Delta = 50$ as before. We test the attack with different hard limits, $C = 2.5, 5, 7.5$. The results in Table 7 show that, in general, the attack can make the agent take actions closer to the target actions with more corruption, implying that our attack can efficiently utilize the corruption budget. We also notice that when C is already abundant, the value of ϵ decreases slower as C increases. The reason is that the value of C is already high enough, so the attacker rarely needs to stop applying corruption during training. In this case, to utilize a high budget of C, the attacker can increase the value of Δ to achieve a smaller value of ϵ.

Alg-$\frac{C}{T}$	Clean ($\frac{\epsilon}{T}$)	Adapt ($\frac{\epsilon}{T}$)
DDPG-7.5	0.52(0.05)	0.10(0.01)
TD3-7.5	0.51(0.07)	0.11(0.02)
SAC-7.5	0.52(0.02)	0.11(0.02)
DDPG-5	0.52(0.05)	0.09(0.00)
TD3-5	0.51(0.07)	0.10(0.02)
SAC-5	0.52(0.02)	0.12(0.01)
DDPG-2.5	0.52(0.05)	0.17(0.03)
TD3-2.5	0.51(0.07)	0.18(0.04)
SAC-2.5	0.52(0.02)	0.17(0.04)

Table 7: Performance of the adaptive target attack for different values of hard limit on budget C.

RL environments with discrete action spaces Here, we empirically verify that the adaptive target attack is efficient against RL environments that have discrete action spaces. In this case, we define the distance measure between two actions as: $d(a_1, a_2) = \mathbb{1}\{a_1 \neq a_2\}$. Under this measure, the distance between two actions is 0 if they are different and 1 if they are identical. we consider two popular classical control problem from Gym (Brockman et al., 2016): MountainCar and Acrobot. For the learning algorithms, we consider the state-of-the-art algorithm: double dueling DQN van Hasselt et al. (2016). For the target policies, we consider three different types of policies, including random, medium, and expert policies. We test the attack in the scenario where there is no constraint on the total corruption budget C. The results are shown in Table 8. Similar to the case of continuous action spaces, the attack can make the agent mostly take exactly the target actions during training with very limited attack budgets. The empirical observation verifies that the adaptive target attack is also efficient against the RL environments with discrete action space.

Table 8: Efficiency of the adaptive target attack in the RL environments with discrete action spaces.

Env-Tar-Alg-Δ	ϵ (clean)	ϵ (adaptive)	C (adaptive)
Acrobot-rand-double-5	0.68 ± 0.01	0.02 ± 0.01	0.10 ± 0.05
Acrobot-med-double-5	0.36 ± 0.01	0.08 ± 0.01	0.40 ± 0.05
Acrobot-exp-double-5	0.33 ± 0.01	0.09 ± 0.01	0.45 ± 0.05
MountainCar-rand-double-5	0.64 ± 0.01	0.03 ± 0.005	0.15 ± 0.02
MountainCar-med-double-5	0.33 ± 0.01	0.07 ± 0.02	0.35 ± 0.10
MountainCar-exp-double-5	0.23 ± 0.01	0.07 ± 0.02	0.35 ± 0.10

C Robot Behavior Controlled by Different Policies

In this section, we show the simulation results of a robot controlled by different policies in the HalfCheetah environment. A detailed explanation of the environment can be found in Todorov et al. (2012); Wawrzyński (2009). In general, the cat-like robot (cheetah) is controlled to run on a flat terrain. The agent can apply torques at the conjunctions of the cheetah, and the goal is to make the cheetah run as fast as possible.

First, we show the motion of the cheetah controlled by an expert-level policy. The policy is learned by training the DDPG algorithm on the actual environment for a sufficient amount of time. In Fig 2, we show sample frames of the motion focusing on the moment when the front leg of the cheetah touches the ground. The full video can be found in the supplementary materials. We observe that the Cheetah runs fast and elegantly, as expected. Particularly, whenever its front tip touches the ground, the tip is usually almost vertical to the ground. After the contact, the front tip swings backward. This is similar to how a horse runs in nature.

Next, we show that the motion of the cheetah is controlled by a medium policy, which is also the target policy we use in our experiments. The policy is learned by the TD3 algorithm training on the actual environment for an insufficient time. In this case, we observe IN Fig 2 that the Cheetah runs slower than the expert policy. In addition, sometimes when the front tip of the cheetah touches the ground, it is nearly horizontal to the ground. Even more, afterward, the front leg almost immediately swings forward. This is very different from the features of the expert policy case. Such an unnatural running behavior could be why the cheetah runs slower and might cause damage to the robot in the long run in practice.

In the end, we show the motion of the cheetah controlled by the policy learned under the attack. Specifically, we examine the policy learned in our experiment in Section 4.2, where the target policy is set as the medium policy we show above, and the attacker applies the adaptive target attack. Although the learned policy is not exactly the same as the medium policy, we observe in Fig 2 that the cheetah controlled by the learned policy exhibits the same behavior as that controlled by the medium policy. This suggests that the adaptive target attack successfully teaches the agent a dangerous behavior with a limited attack budget.

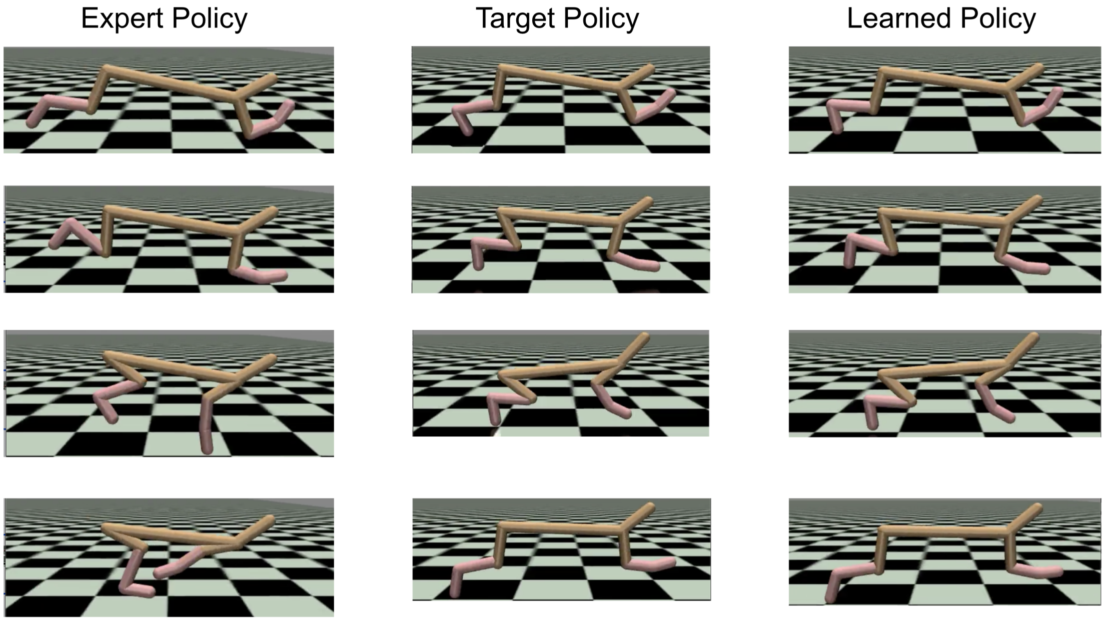

Figure 2: The motion of the cheetah when controlled by different policies. The 'expert policy' is learned by the agent when there is no attack. The 'target policy' is learned by the DDPG algorithm training for sufficient time with no attack. The target policy is the medium policy we use in our experiments in Section 4. It is learned by the TD3 algorithm training for an insufficient time. The 'learned policy' is learned by the DDPG algorithm under the adaptive target attack with the 'target policy' as the target. For each policy, we sample four frameworks before and after the front tip of the cheetah touches the ground. The contact happens between the second and the third frames.

D Discussion on Attack Detectability

A practical consideration for robustness against attack is detecting the abnormality in the training data induced by poisoning attacks. We find a line of literature studying the problem of detecting the anomaly during the testing time when the learned policy is deployed in the context of RL Müller et al. (2022); Haider et al. (2023); Nasvytis et al. (2024). These works assume that the training time is uncorrupted and train their detector during the training process, which cannot be used to detect corruption during training. Therefore, it is unclear how to defend against the data poisoning attack through anomaly detection at training time in the online RLHF setting. Particularly, we find this challenging for the following reasons.

1. The distribution of the data observed in online training keeps changing during training. As a result, it is natural to see 'abnormal' data points compared to the current dataset distribution. For example, consider an agent that is initialized from a random policy and eventually converges to an expert policy during a typical online RL training process. The distribution of the data at the beginning is collected by random behavior and has low rewards. As training continues, the agent's policy improves and starts collecting high rewards, which are 'abnormal' compared to the past reward distribution. How the data distribution changes depends on the environment and the learning algorithm, and it is unclear how it would change before training.

2. In an RL task, the agent visits different states sequentially, and at each state, it collects a reward from a reward function depending on the action it takes. In different states, the reward as a function of action can be very different. It is normal to collect high rewards at a state and low rewards at a different state using the same exploration policy.

In conclusion, it is challenging to judge if an 'abnormal' data point is induced by the attack because they naturally exist in the online RL training process.

Empirically, we test an intuitive anomaly detection method for the dynamic data distribution. The method keeps track of the most recent N rewards collected during training. Let μ^t and σ^t be the mean and variance of the past N rewards before timestep t, the reward r^t received at time t is detected as abnormal if $|r^t - \mu^t| \geq \alpha \cdot \sigma^t$ where $\alpha > 0$ is a hyper-parameter set by the method. We set a moderate window size $N = 500$ and test on the Halfcheetah environment learned by the DDPG algorithm. In the case when there is no attack, in order not to detect true rewards as abnormal data, the method needs to set $\alpha > 100$. Under this setup, when the agent is under our adaptive target attack with $\Delta = 50$ (the setup in our main results), the method cannot detect any corrupted data. We have a similar observation when we set the size of the window to be different values $N = 100, 1000$. The result suggests that it can be challenging to detect data corruption by anomaly detection in the online RL setting. We believe it is a challenging yet important future work to develop efficient anomaly detection methods for online RL during training.

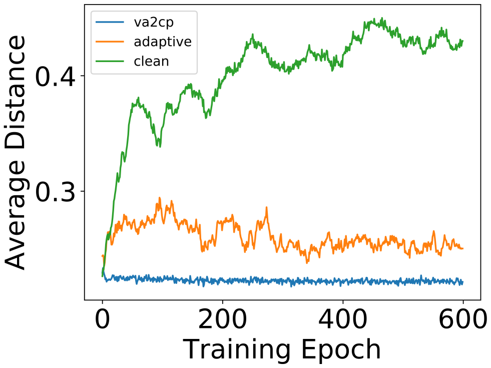

Figure 3: Training log of PPO algorithm learned in Swimmer environment under the attacks.

E Comparison to va2c-p

We compare our attack to the va2c-p attack from Sun et al. (2020). As an example, we consider the Swimmer environment learned by the PPO algorithm. To avoid degrading the effect of the va2c-p attack, we implement our attacks in their code and run the experiment. Due to the limitation in their code, only naive target policies are considered which always output the same action for any state. We adapt our attack to work with the constraints in their work by forcing the attack to stop applying perturbation if doing so breaks the constraint. For the values of constraints, we set $\epsilon = 0.5$ and $C/K = 1$. For our adaptive target attack, we set $\Delta = 1$. The result of the training log of the agent under the attacks is shown in Figure 3. We find that our attack has comparable efficiency to the va2c-p attack. Note that the comparison is not fair in the first place since va2c-p attack works in white-box setting while our attack works in black-box setting. Our attack is also modified here as it is optimized for a different constraint on the attack that is actually more general. We also find that our attack runs about 50 times faster than the va2c-p attack. It is noteworthy that our attack is also much easier to implement as its implementation is independent of the learning algorithm and only depends on the interactions during training.