VULNERABILITY-AWARE POISONING MECHANISM FOR ONLINE RL WITH UNKNOWN DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Poisoning attacks on Reinforcement Learning (RL) systems could take advantage of RL algorithm's vulnerabilities and cause failure of the learning. However, prior works on poisoning RL usually either unrealistically assume the attacker knows the underlying Markov Decision Process (MDP), or directly apply the poisoning methods in supervised learning to RL. In this work, we build a generic poisoning framework for online RL via a comprehensive investigation of heterogeneous poisoning models in RL. Without any prior knowledge of the MDP, we propose a strategic poisoning algorithm called Vulnerability-Aware Adversarial Critic Poison (VA2C-P), which works for most policy-based deep RL agents, closing the gap that no poisoning method exists for policy-based RL agents. VA2C-P uses a novel metric, stability radius in RL, that measures the vulnerability of RL algorithms. Experiments on multiple deep RL agents and multiple environments show that our poisoning algorithm successfully prevents agents from learning a good policy or teaches the agents to converge to a target policy, with a limited attacking budget.

1 INTRODUCTION

Although reinforcement learning (RL), especially deep RL, has been successfully applied in various fields, the security of RL techniques against adversarial attacks is not yet well understood. In real-world scenarios, including high-stakes ones such as autonomous driving vehicles and healthcare systems, a bad decision may lead to a tragic outcome. Should we trust the decision made by an RL agent? How easy is it for an adversary to mislead the agent? These questions are crucial to ask before deploying RL techniques in many applications.

In this paper, we focus on *poisoning attacks*, which occur during the training and influence the learned policy. Since training RL is known to be very sample-consuming, one might have to constantly interact with the environment to collect data, which opens up a lot of opportunities for an attacker to poison the training samples collected. Therefore, understanding poisoning mechanisms and studying the vulnerabilities in RL are crucial to provide guidance for defense methods. However, existing works on adversarial attacks in RL mainly study the test-time *evasion attacks* Chen et al. (2019) where the attacker crafts adversarial inputs to fool a well-trained policy, but does not cause any change to the policy itself. Motivated by the importance of understanding RL security in the training process and the scarcity of relevant literature, in this paper, we *investigate how to poison RL agents and how to characterize the vulnerability of deep RL algorithms.*

In general, RL agents learn in an "online" manner. At every iteration, the agent rolls out experience from the environment with its policy, and uses the experience to improve its policy, then uses the new policy to roll out new experience, etc. Poisoning in online RL is significantly different from poisoning in classic supervised learning (SL), even online SL, and could be more difficult due to the following challenges.

Challenge I – Future Data Unavailable in Online RL. Poisoning approaches in SL (Muñoz-González et al., 2017; Wang & Chaudhuri, 2018) usually require the access to the whole training dataset, so the attacker can decide the optimal poisoning strategy before the learning starts. However, in online RL, the training data (trajectories) are generated by the agent while it is learning. Although the optimal poison should work in the long run, the attacker can only access and change the data in the current iteration, since the future data is not yet generated.

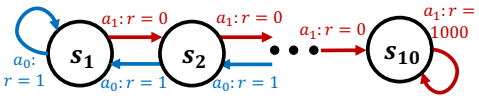

Figure 1: An example of difficult poisoning.

Challenge II – Data Samples No Longer i.i.d.. It is well-known that in RL, data samples (state-action transitions) are no longer i.i.d. due to the Markovian property, which makes learning challenging, since we should consider the long-term reward rather than the immediate result. However, we notice that data samples being not i.i.d. also makes poisoning attacks challenging. For example, an attacker wants to reduce the agent's total reward in a task shown as Figure 1; at state s_1, the attacker finds that a_1 is less rewarding than a_0; if the attacker only looks at the immediate reward, he will lure the agent into choosing a_1. However, following a_1 finally leads the agent to s_{10} which has a much higher reward.

Challenge III – Unknown Dynamics of Environment. Although Challenge I and II can be partially addressed by predicting the future trajectories or steps, it requires prior knowledge on the dynamics of the underlying MDP. Many existing poisoning RL works (Rakhsha et al., 2020; Ma et al., 2019) assume the attacker has perfect knowledge of the MDP, then compute the optimal poisoning. However, in complex environments such as Atari games, knowing the dynamics of the MDP is difficult. Although the attacker could potentially interact with the environment to build an estimate of the environment model, the cost of interacting with the environment could be unrealistically high, market making (Spooner et al., 2018) for instance. In this paper, we study a more realistic scenario where the attacker does not know the underlying dynamics of MDP, and can not directly interact with the environment, either. Thus, the attacker learns the environment only from the agent's experience.

In this paper, we systematically investigate poisoning in RL by considering all the aforementioned RL-specific challenges. Previous works either do not address any of the challenges or only address some of them. Behzadan & Munir (2017) achieve policy induction attacks for deep Q networks (DQN). However, they treat output actions of DQN similarly to labels in SL, and do not consider Challenge II that the current action will influence future interactions. Ma et al. (2019) propose a poisoning attack for model-based RL, but they suppose the agent learns from a batch of given data, not considering Challenge I. Rakhsha et al. (2020) study poisoning for online RL, but they require perfect knowledge of the MDP dynamics, which is unrealistic as stated in Challenge III.

Summary of Contributions. (1) We propose a practical poisoning algorithm called Vulnerability-Aware Adversarial Critic Poison (VA2C-P) that works for deep policy gradient learners without any prior knowledge of the environment. To the best of our knowledge, VA2C-P is *the first practical algorithm that poisons policy-based deep RL methods*. **(2)** We introduce a novel metric, called stability radius, to characterize the stability of RL algorithms, measuring and comparing the vulnerabilities of RL algorithms in different scenarios. **(3)** We conduct a series of experiments for various environments and state-of-the-art deep policy-based RL algorithms, which demonstrates RL agents' vulnerabilities to even weaker attackers with limited knowledge and attack budget.

2 Related Work

The main focus of this paper is on poisoning RL, an emerging area in the past few years. We survey related works of adversarial attacks in SL and evasion attacks in RL in Appendix A, as they are out of the scope of this paper.

Targeted Poisoning Attacks for RL. Most RL poisoning researches work on targeted poisoning, also called policy teaching, where the attacker leads the agent to learn a pre-defined target policy. Policy teaching can be achieved by manipulating the rewards (Zhang & Parkes, 2008; Zhang et al., 2009) or dynamics (Rakhsha et al., 2020) of the MDP. However, they require the attackers to not only have prior knowledge of the environments (e.g., the dynamics of the MDP), but also have the ability to *alter the environment* (e.g. change the transition probabilities) , which are often unrealistic or difficult in practice.

Poisoning RL with Omniscient Attackers. Most guaranteed poisoning RL literature (Rakhsha et al., 2020; Ma et al., 2019) assume *omniscient attackers*, who not only know the learner's model, but also know the underlying MDP. However, as motivated in the introduction, the underlying MDP is usually either unknown or too complex in practice. Some works poison RL learners by changing the reward signals sent from the environment to the agent. For example, Ma et al. (2019) introduce a policy teaching framework for batch-learning model-based agents; Huang & Zhu (2019) propose a reward-poisoning attack model, and provide convergence analysis for Q-learning; Zhang et al.

(2020) present an adaptive reward poisoning method for Q-learning (while it also extends to DQN) and analyze the safety thresholds of RL; these papers all assume the attacker knows not only the models of the agent, but also the parameters of the underlying MDP, which could be possible in a tabular MDP, but hard to realize in large environments and modern deep RL systems.

On the contrary, *We consider non-omniscient attackers* who do not know the underlying MDP or environment in this paper. The non-omniscient attackers can be further divided into two categories: *white-box* attackers, who know the learner's model, and *black-box* attackers, who do not know the learner's model. They both tap the interactions between the learner and the environment.

Black-box Poisoning for Value-based Learners. Although there are many successful black-box evasion approaches (Xinghua et al., 2020; Inkawhich et al., 2020), black-box poisoning in RL is rare. There is a *black-box* attacking method for a value-based learner (DQN) proposed by Behzadan & Munir (2017), which does not require the attacker to know the learner's model or the underlying MDP. In this work, the attacker induces the DQN agent to output the target action by perturbing the state with Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) in every step. However, the data-correlation problem of RL (Challenge II) is not considered, and FGSM attack does not work for policy-based methods due to their high stochasticity, as we show in experiments.

In this paper, we propose a new poisoning algorithm for *policy-based deep RL* agents, which can achieve *both non-targeted and targeted* attacks. We do not require any prior knowledge of the environment. And our algorithm works not only when the attacker knows the learner's model (white-box), but also when the leaner's model is hidden (black-box).

3 Problem Formulation for Poisoning Online RL

3.1 Notations and Preliminaries

In RL, an agent interacts with the environment by taking actions, observing states and receiving rewards. The environment is modeled by a Markov Decision Process (MDP), which is denoted by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition kernel, R is the reward function, $\gamma \in (0, 1)$ is the discount factor, and μ is the initial state distribution. A *trajectory* $\tau \sim \pi$ generated by *policy* π is a sequence $s_1, a_1, r_1, s_2, a_2, \cdots$, where $s_1 \sim \mu$, $a_t \sim \pi(a|s_t)$, $s_{t+1} \sim P(s|s_t, a_t)$ and $r_t = R(s_t, a_t)$. The goal of an RL agent is to find an optimal policy π^* that maximizes the *expected total rewards* η, which is defined as $\eta(\pi) = \mathbb{E}_{\tau\sim\pi}[r(\tau)] = \mathbb{E}_{s_1,a_1,\cdots\sim\mu,\pi,P,R}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$.

We use an overhead check sign ˇ on a variable to denote that the variable is poisoned. For example, if the attacker perturbs a reward r_t, then the poisoned reward is denoted as $\check{r}_t$. If a policy π is updated with poisoned observation, then it is denoted as $\check{\pi}$.

3.2 A Unified Formulation for Poisoning Online Policy-based RL.

Procedure of Online Learning. We consider a classical online policy-based RL setting, where the learner iteratively updates its *policy* π parametrized by θ, through K iterations with the environment. For notation simplicity, we omit θ and use π_k to denote π_{θ_k}, the learner's policy at iteration k. The online learning process is described as below.

At iteration $k = 1, \cdots, K$,
(1) The agent uses the current policy π_k to roll out **observation** $\mathcal{O}_k = (\mathcal{O}_k^{\boldsymbol{s}}, \mathcal{O}_k^{\boldsymbol{a}}, \mathcal{O}_k^{\boldsymbol{r}})$ from environment $\mathcal{M}_k$, where $\mathcal{O}_k^{\boldsymbol{s}} = [s_1, s_2, \cdots], \mathcal{O}_k^{\boldsymbol{a}} = [a_1, a_2, \cdots], \mathcal{O}_k^{\boldsymbol{r}} = [r_1, r_2, \cdots]$ are respectively the sequence of states, actions and rewards generated at iteration k.
(2) The agent updates its policy parameters θ with its algorithm f. Most policy-based algorithms perform *on-policy* updating, i.e., update policy only by the current observation $\mathcal{O}_k$. The on-policy update can be then formalized as $\pi_{k+1} = f(\pi_k, \mathcal{O}_k) \approx \text{argmax}_\pi J(\pi, \pi_k, \mathcal{O}_k)$, where J is an objective function defined by algorithm f, e.g., the expected total reward $\eta(\pi)$.

Procedure of Online Poisoning. A poisoning attacker influences the learner in the training process by perturbing the training data. In SL, the training data consists of data features and labels, and the attacker poisons the training data before the learning starts. In contrast, the training data in RL is the observation a learner gains from the environment, i.e., $\mathcal{O} = (\mathcal{O}^{\boldsymbol{s}}, \mathcal{O}^{\boldsymbol{a}}, \mathcal{O}^{\boldsymbol{r}})$. At iteration k, the attacker taps the observation $\mathcal{O}_k$ returned from the environment, and can poison it into $\check{\mathcal{O}}_k$, then

send $\check{\mathcal{O}}_k$ to the learner before policy updating. Figure 2 illustrates this online learning-poisoning procedure, where we can see that learning and poisoning are convoluted and inter-dependent. [1]

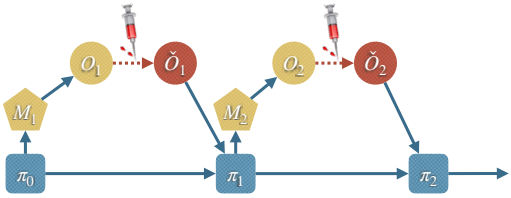

Figure 2: The online poisoning-learning process.

Attacker's Poison Aim. The attacker has multiple candidate *poison aims* – it may attack the observed states $\mathcal{O}^{\boldsymbol{s}}$, the executed actions $\mathcal{O}^{\boldsymbol{a}}$ or the received rewards $\mathcal{O}^{\boldsymbol{r}}$. Therefore, the attacker's poison aim $\mathcal{D}$ is in $\{\mathcal{O}^{\boldsymbol{s}}, \mathcal{O}^{\boldsymbol{a}}, \mathcal{O}^{\boldsymbol{r}}\}$. For example, an RL agent recommends items to users in a recommender system. The RL agent chooses an item (i.e., an action) for a user (i.e., a state), and the user may or may not choose to click on the recommended item (i.e., a reward). An adversary might manipulate the reward (poisoning $\mathcal{D} = \mathcal{O}^{\boldsymbol{r}}$); for instance blocking the user's click from the RL agent or creating a fake click. An adversary might also alter the state (poisoning $\mathcal{D} = \mathcal{O}^{\boldsymbol{s}}$); for instance raising a teenager user's age which could result in inappropriate recommendations. Moreover, an adversary might also change the action (poisoning $\mathcal{D} = \mathcal{O}^{\boldsymbol{a}}$); such as inserting a fake recommendation into the agent's list of recommendations.

A Poisoning Framework for RL. We focus on proposing a poisoning mechanism for the above challenging online learning scenario. We first formalize the poisoning attacking at iteration k as a *sequential bilevel optimization* problem in Problem (Q), and explain the details of the problem in the remaining of this section.

$$
\begin{aligned}
\underset{\check{\mathcal{D}}_k, \cdots, \check{\mathcal{D}}_K}{\operatorname{argmin}} \quad & \sum\nolimits_{j=k}^{K} \lambda_j L_A(\check{\pi}_{j+1}) && \text{((a) attacker's weighted loss)} \qquad \text{(Q)} \\
s.t. \quad & \check{\pi}_{j+1} = \operatorname{argmax}_{\pi} J(\pi, \tilde{\pi}_j, \check{\mathcal{O}}_j | \check{\mathcal{D}}_j), \forall k \leq j \leq K && \text{((b) imitate the learner)} \\
& \sum\nolimits_{j=1}^{K} \mathbf{1}\{\check{\mathcal{D}}_j \neq \mathcal{D}_j\} \leq C && \text{((c) limited-budget)} \\
& U(\mathcal{D}_j, \check{\mathcal{D}}_j) \leq \epsilon, \forall 1 \leq j \leq K && \text{((d) limited-power)}
\end{aligned}
$$

(a) Attacker's Weighted Loss. $L_A(\check{\pi})$ measures the attacker's loss w.r.t. a poisoned policy π. As the definition of poisoning implies, the attacker influences or misleads the learner's policy. $\lambda_{k:K}$ are the weights of future attacker losses, controlling how much the attacker value the poisoning results in different iterations. The goal of the attacker is either *(1) non-targeted poisoning, which minimizes the expected total rewards of the learner, i.e.,* $L_A = \eta(\check{\pi})$, or *(2) targeted poisoning, which induces the learner to learn a pre-defined target policy, i.e.,* $L_A = \text{distance}(\check{\pi}, \pi^\dagger)$, where $\text{distance}(\check{\pi}, \pi^\dagger)$ can be any distance measure between a learned policy $\check{\pi}$ and a target policy $\pi^\dagger$. Note that the targeted poisoning objective can usually be directly computed with a prior target policy, while non-targeted poisoning has a "reward-minimizing" objective, which is the reverse of the learner's objective. Without any prior knowledge of the environment, non-targeted poisoning is usually more difficult, as the attacker needs to first learn "what is the worst way" (which is as difficult as a learning problem by a RL agent) and then lead the learner to that way (which is as difficult as a targeted poisoning problem, assuming leading an agent to different policies is roughly equally challenging). However, most existing poisoning work focus on targeted poisoning, while reward-minimizing may deprave the policy without having any pre-defined target policy. Thus in this paper, we make more efforts to solve the reward-minimizing poisoning problem.

(b) Imitate the Policy-Based Learner. To confidently mislead a learner, the attacker needs to predict how the learner will behave under the poison. In the equation "imitate the learner", $\mathcal{D} \in \{\mathcal{O}^{\boldsymbol{s}}, \mathcal{O}^{\boldsymbol{a}}, \mathcal{O}^{\boldsymbol{r}}\}$ stands for the poison aim of the poisoning. $\tilde{\pi}_j$ refers to the attacker's imitation of the learner's policy at j-th iteration, and $\operatorname{argmax}_{\pi} J(\pi, \tilde{\pi}_j, \check{\mathcal{O}}_j | \check{\mathcal{D}}_j)$ is the learner's objective function defined by its algorithm f. The attacker imitates the poisoned learning process to predict what next-policy the learner can get, which is dependent on the *attacker's knowledge*. As introduced in Section 2, we deal with both white-box and black-box attackers, and both of them do not know the environment $\mathcal{M}$. A *white-box* attacker knows the current and past observations $\mathcal{O}_{1:k}$, the learner's algorithm f and policy π, so it can directly copy the policy $\tilde{\pi}_j = \pi_j, \forall j$. A *black-box* attacker only

[1] In this paper, we assume the attacker poisons the observation, which is the most universal setting in practice. Appendix B extends our problem formulation to a more general case where the special environment poisoning is incorporated.

knows the current and past observations $\mathcal{O}_{1:k}$, and may know the learner's algorithm f, but does not know the learner's policy π. In this case, the attacker has to guess what π could be at every iteration. Section 4.3 states how to guess π.

(c,d) Limited-budget and Limited-power. In practice, the ability of an attacker is usually restricted by some constraints. For the online poisoning problem, we consider attacker's constraints in two forms: (1) (*attack budget* C) the total number of iterations that the attacker could poison does not exceed C; (2) (*attack power* ϵ) in one iteration, the total change[2] $U(\mathcal{D}_k, \check{\mathcal{D}}_k)$ between $\mathcal{D}_k$ and $\check{\mathcal{D}}_k$ can not be larger than ϵ. Attack power controls the amount of perturbation, as commonly used in the adversarial learning literature. Attack budget considers the frequency of attack, which is similar to the constraint studied by Wang & Chaudhuri (2018).

Problem (Q) is a generic formulation, covering a variety of poisoning models. and specifies the best poison an attacker can execute. However, directly solving Problem (Q) is prohibitive, as (1) the future observations $\mathcal{O}_{k+1:K}$ are unknown when poisoning the k-th iteration, as the attacker has no knowledge of the underlying MDP. (2) the limited-budget constraint is analogous to ℓ_0-norm regularization, which is generally NP-hard (Nguyen et al., 2019); and (3) minimizing attacker's loss while maximizing learner's gain is non-convex minimax optimization, which is a complex problem (Perolat et al., 2015).

In spite of the above difficulties, we introduce a practical method to approximately and effectively solve Problem (Q) in Section 4.

4 VA2C-P: Poison Policy Gradient Learners

In this section, we propose a practical and efficient poisoning algorithm called Vulnerability-Aware Adversarial Critic Poison (VA2C-P) for policy gradient learners. Without loss of generality, we assume the loss weights $\lambda_j = 1$ for all $j = 1, \cdots, K$.

Main Idea. As discussed in Section 3.2, Problem (Q) is difficult mainly because of the unknown future observations and the limited budget constraint. In other words, it is hard to exactly determine (1) what kind of attack benefits the future the most, and (2) which iterations are worth attacking the most. Thus, we propose to break Problem (Q) into two decisions: when to attack, and how to attack. The "when to attack" decision allocates the limited budget to iterations which are more likely to be influenced by the attacker, and the "how to attack" decision utilizes limited power to minimize the attacker's loss. We introduce two mechanisms of VA2C-P, vulnerability-awareness and adversarial critic, to make these two decisions respectively.

4.1 Decision 1: When to Attack – Vulnerability-Aware

To answer the question of when to attack, we identify the iterations under which the learner's policy gets more depraved by the same level of attack power. Inspired by the notion of stability in learning theory, which measures how a machine learning algorithm changes due to a small perturbation of the input data, we formally investigate the stability of an RL algorithm, which is the first attempt in the existing literature to the best of our knowledge.

Stability of RL Algorithms. We first focus on a single update process of an algorithm f. An update $\pi' = f(\pi, \mathcal{O})$ is stable if a limited-power poisoning attack does not cause any difference on the output policy π'. That is, the learning algorithm produces the same result regardless of the presence of the poison. More formally, we define the concept of *stability radius of one update* in Definition 1.

Definition 1 (Stability Radius of One Update)**.** [3] *For the update of an RL algorithm $\pi' = f(\pi, \mathcal{O})$, with any poison aim $\mathcal{D}$, the δ-stability radius of the update is defined as the minimum poison power needed to cause δ change in policy (called δ-policy-discrepancy)*

$$\phi_{\delta,\mathcal{D}}(f, \pi, \mathcal{O}) = \inf_{\epsilon}\{\exists \check{\mathcal{D}} \text{ s.t. } U(\mathcal{D}, \check{\mathcal{D}}) \leq \epsilon \text{ and } d^{\max}[\pi' || \check{\pi}'] > \delta, \text{ where } \check{\pi}' = f(\pi, \check{\mathcal{O}}|\check{\mathcal{D}})\}, \quad (1)$$

Policy discrepancy $d^{\max}[\pi_1||\pi_2] = \max_s d\big[\pi_1(\cdot|s)||\pi_2(\cdot|s)\big]$, where $d[\cdot||\cdot]$ could be any measure of distribution distance.

[2] There are many choices of $U(\cdot,\cdot)$. For example, the total effort w.r.t. $\mathcal{O}^{\boldsymbol{s}}$-poisoning can be the average ℓ_p-distance between any poisoned and unpoisoned state in $\mathcal{O}^{\boldsymbol{s}}$ and $\check{\mathcal{O}}^{\boldsymbol{s}}$.

[3] In order to characterize the stability and robustness of RL algorithms in a principled way, we provide more measures for the vulnerability of RL in both training time and test time. See Appendix D.2) for details.

Remarks. (1) The one-update stability radius is w.r.t. the algorithm f, the old policy π, the clean observation $\mathcal{O}$ and the poison aim $\mathcal{D}$. (2) Poison with power under $\phi_{\delta,\mathcal{D}}(f,\pi,\mathcal{O})$ will not cause the policy distributions to change more than δ. (3) As shown by Proposition 2 in Appendix D.1, poison with power under $\phi_{\delta,\mathcal{D}}(f,\pi,\mathcal{O})$ will not make the reward drop more than $O\big(\delta^2\gamma(1-\gamma)^{-2}\big)$.

Stability radius measures the minimal effort needed to make the poisoned policy $\check{\pi}$ notably different from the *clean next-policy* π' which the learner will get if no poison is applied. An attacker wants to utilize its limited effort to influence the learner as much as possible. Therefore, the idea of vulnerability-aware attack is to estimate the vulnerability of each update and attack the most vulnerable ones. More specifically, if the attacker finds an ϵ-powered attack results in a policy discrepancy larger than some threshold δ, then it can conclude $\phi_\delta(f,\pi,\mathcal{O}) \leq \epsilon$, and the current update is relatively vulnerable. In practice, we have the fixed budget C instead of δ, so we perform the vulnerability check in an adaptive way: attacking the C iterations where ϵ-powered attacks can trigger the highest policy discrepancies. Section 4.3 introduces an algorithm to realize this idea.

4.2 Decision 2: How to Attack – Adversarial Critic

Since "when to attack" decision tackles the limited-budget constraint, now we turn our attention to minimizing the attacker's loss while satisfying the limited-power constraint. If the attacker has already decided to poison the k-th iteration due to its high vulnerability, then the decision of how to attack should be made before the learner uses the observation to update the policy. We relax the original Problem (Q) into Problem (P) as below.

$$\begin{aligned} &\operatorname{argmin}_{\check{\mathcal{D}}_k} && L_A(\check{\pi}_{k+1}) \\ &s.t. && \check{\pi}_{k+1} = \operatorname{argmax}_\pi J(\pi, \tilde{\pi}_k, \check{\mathcal{O}}_k | \check{\mathcal{D}}_k) \\ & && U(\mathcal{D}_k, \check{\mathcal{D}}_k) \leq \epsilon \end{aligned} \tag{P}$$

Compared with Problem (Q), Problem (P) does not consider future losses, which require the unavailable future observations. Instead, Problem (P) finds a greedy attack $\check{\mathcal{D}}_k$ to minimize the loss of the immediate next iteration. The solution to Problem (P) is always feasible to Problem (Q, although might not be optimal. Appendix F.2 provides details about the optimality of Problem (Q).

In practice, it is also challenging to estimate $L_A(\check{\pi})$. For targeted attacking, $L_A = \text{distance}(\check{\pi}, \pi^\dagger)$ is directly computable with a properly defined distance metric, but for non-targeted attacking, the loss $L_A = \eta(\check{\pi})$ can not be directly computed, since $\check{\pi}$ is not the behavior policy. Although one can use importance sampling to evaluate $\eta(\check{\pi})$ with the current trajectories generated by the learner's policy, i.e., $\mathbb{E}_{\tau\sim\pi_{k-1}}[\frac{\pi_k(\tau)}{\pi_{k-1}(\tau)} r(\tau)]$, it may suffer from a high variance (Schulman et al., 2015b) when there are few trajectories. To solve this challenge, we introduce another mechanism, adversarial critic.

Adversarial Critic. Inspired by the Actor-Critic method, we propose to let the attacker learn a value function (network) $\tilde{V}_\omega$ with observations of the learner, and use $\tilde{V}_\omega$ to direct the learner to the decreasing-value direction, which is called *Adversarial Critic*. Note that because of poisoning, the value network (if any) held by the learner usually fails to fit the real value of its policy, but the attacker is able to learn the real values to make the attacking stronger. Then, the attacker can use $\tilde{V}_\omega$ to evaluate the true value of a poisoned policy. In this case, the loss becomes $\mathbb{E}_{s,a\sim\pi_k}[\frac{\check{\pi}(a|s)}{\pi_k(a|s)}\big(G(s_t,a_t) - \tilde{V}_\omega(s_t)\big)]$, where G is the discounted future reward $\sum_{i=t}^{T}\gamma^{i-t} r_t$.

4.3 Poisoning Algorithm VA2C-P.

The implementation details are illustrated in Algorithm 2 in Appendix E, and the main steps are summarized as below.

Suppose c iterations are already poisoned, at iteration k, with poison aim $\mathcal{D}$:
Step 1: update the adversarial critic $\tilde{V}_\omega$ with the current observation $\mathcal{O}_k$;
Step 2: imitate learner's update with clean observation $\mathcal{O}_k$ and get the clean next-policy π'_{k+1};
Step 3: find the greedy poisoning $\check{\mathcal{D}}_k$ by solving Problem (P) with power ϵ and critic $\tilde{V}_\omega$;
Step 4: compute $\check{\pi}_{k+1}$ by imitating the learner with $\check{\mathcal{D}}_k$;
Step 5: estimate the policy discrepancy $\widehat{\psi}_k$ between π'_{k+1} and $\check{\pi}_{k+1}$;
Step 5: save $\widehat{\psi}_k$ to the pool of all historical policy discrepancies $\Psi \leftarrow \Psi \cup \widehat{\psi}_k$;

Step 6: send the learner the poisoned observation if $\widehat{\psi}_k \geq \lfloor \frac{C-c}{K-k} \rfloor$-th largest element in Ψ; otherwise send the clean observation;
Step 7: $k \leftarrow k+1$, repeat.

In this algorithm, We first find the best direction for a greedy attack with the current attack power ϵ, then estimate how much the output policy will change by computing the policy discrepancy between the clean next-policy and the poisoned next-policy. Then the attacker poisons if the policy discrepancy ranks high in the historical policy discrepancies, via an unbiased estimation of the rank (Step 6). In Step 2-4, the attacker needs to imitate the learner's update with clean observation and poisoned observation. As Section 3 states, the imitation step requires $\tilde{\pi}$, attacker's approximation of the learner's policy. For a white-box attacker, $\tilde{\pi}_k = \pi_k, \forall k$, while a black-box attacker can train a *pseudo learner* by taking the same (poisoned) observations as the learner, i.e., $\tilde{\pi}_k = f(\tilde{\pi}_{k-1}, \check{\mathcal{O}}_{k-1})$. As claimed and verified by many black-box attacking methods (Behzadan & Munir, 2017), adversarial attacks are usually transferable, i.e., if the attack works on the pseudo learner, then the attack would also work for the real learner. In Step 3, we use projected gradient descent to search for the optimal $\check{\mathcal{D}}$. And the computation details are illustrated in Appendix E.

5 Experiments

In this section, we evaluate the performance of VA2C-P by poisoning multiple algorithms on various environments. We demonstrate that VA2C-P can effectively reduce the total reward of a training agent, or force the agent to choose a specific policy with limited power and budget. Moreover, VA2C-P works for heterogeneous poison aims, and works in both white-box and black-box settings.

Experiment Setup. We choose 4 policy-gradient learning algorithms, including Vanilla Policy Gradient (Sutton et al., 2000), A2C (Mnih et al., 2016), ACKTR (Wu et al., 2017) and PPO (Schulman et al., 2017). And we choose 5 Gym (Brockman et al., 2016) environments with increasing difficulty levels: CartPole, LunarLander, Hopper, Walker and HalfCheetah. All results are averaged over 10 random seeds. The total effort U is calculated by the normalized ℓ_2-distance. See Appendix G.1 for the expression of U, as well as more hyper-parameter settings.

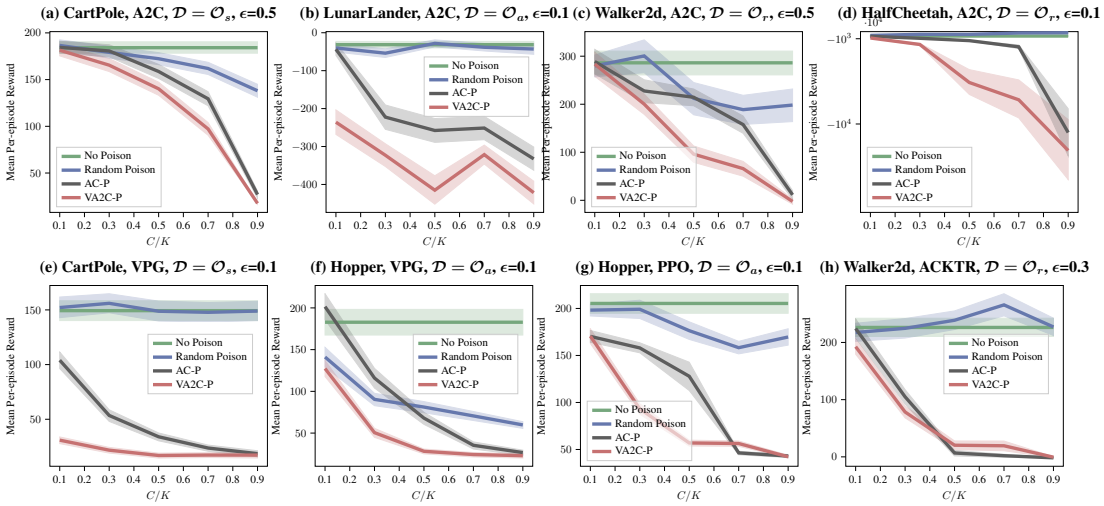

Figure 3: Comparison of mean per-episode reward gained by VPG, PPO, A2C, ACKTR on various environments, under no poisoning, random poisoning, AC-P and VA2C-P.

Reward-minimizing Poisoning. *Baselines.* To the best of our knowledge, there is *no existing poisoning algorithm against deep policy-gradient algorithms*. Although some evasion methods (Inkawhich et al., 2020) also work for policy-gradient algorithms, evasion is substantially different from poisoning since it does not influence the policy. Therefore, to show the effectiveness of VA2C-P, we compare it with 3 baselines: (1) the normal learning with no poison; (2) a random attacker which randomly chooses C iterations, and perturbs the reward to an arbitrary direction by ϵ; and (3) a simplified version of our algorithm, called Adversarial Critic Poison (AC-P), which decides "how to attack" in the same way as VA2C-P does, but chooses "when to attack" randomly.

Performance. We first show the reward-minimizing performance of VA2C-P on all three types of poison aims, assuming the attacker knows the learner's model (white-box attack). Figure 3 shows

the rewards of various learners under different kinds of poisoning methods, with different ratios of budget C to the total number of iterations K. Compared with random poisoning, our proposed *VA2C-P and the simplified version AC-P make the reward drop more significantly*, demonstrating the effectiveness of our "how to attack" decisions made by the Adversarial Critic. VA2C-P further outperforms AC-P in almost all cases, which implies that our "when to attack" decisions based on Vulnerability-Awareness work well in practice. An interesting observation is *random poisoning not only does not work well in many cases, but sometimes also facilitates the learner* (Figure 3h). This phenomenon is mainly due to the uncertainty of the environment, as pointed out by *Challenge I* and *Challenge II* in Section 1. Thus, a good poisoning strategy is important.

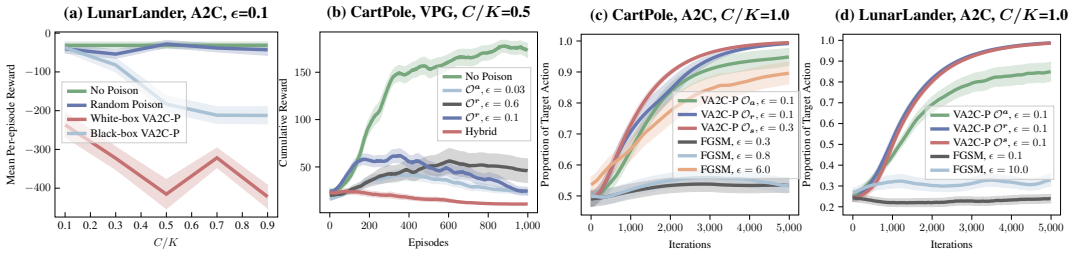

Figure 4: (a) VA2C-P also works under black-box setting; (b) hybrid-aim poisoning could be better than single-aim poisoning; (c)(d) VA2C-P successfully forces the agent to choose the target policy.

Black-box Poisoning. Figure 3 demonstrates the performance of VA2C-P when the attacker knows the learner's model. But when the learner's model is not available (black-box attack), VA2C-P could also work as shown in Figure 4a, where one can see that *black-box poisoning is still effective, although worse than white-box poisoning due to the lack of knowledge*.

Hybrid-aim Poisoning. The experiments shown in Figure 3 assume that the attacker poisons one of the 3 poison aims, $\mathcal{O}^s, \mathcal{O}^a$ or $\mathcal{O}^r$. But if the attacker happens to have access to all of these poison aims, it can do better by performing a "hybrid" poisoning, i.e., at each iteration, evaluate the vulnerability of the algorithm w.r.t. each poison aim with their corresponding attacker power, then choose the one with the highest vulnerability. Figure 4b shows that *adaptive attacking using a hybrid poison aim could significantly outperform attacking using a fixed single poison aim.*

Targeted Poisoning. *Baselines.* Although targeted RL poisoning is studied by many works (Huang & Zhu, 2019; Rakhsha et al., 2020), few of them work for deep RL. Despite that no existing work focuses on deep policy-based methods, we transfer the FGSM-based targeted poisoning method proposed for DQN by Behzadan & Munir (2017) to attack policy-based learners as our baseline. The attacking method is, for each new state, adding a perturbation to it such that the perturbed state is pushed across the decision boundary and towards the target action. See Appendix G.2 for details.

Performance. Figure 4c and 4d respectively show the targeted-attack performance comparison on CartPole and LunarLander, where the target policies are both "always going to the right". The y-axes show the proportion of target actions among all actions taken by the learner. *VA2C-P successfully leads the learner to learn the target policy using all types of poison aims with relatively small attack power.* In contrast, FGSM fails to let the learner select the target action in most cases, even with much larger attack power. The main reason why FGSM works for DQN Behzadan & Munir (2017) but not for policy-gradient methods is, policy-based agents generally learn stochastic policies, and even though FGSM could perturb the state such that the output probability for the target action is 0.51, the agent will still choose other actions with probability 0.49. Therefore, *FGSM cannot easily perform targeted poisoning against policy-based learners.*

We put more experiment results in Appendix G.3, and demo videos in supplementary materials.

6 Conclusion and Discussion

In this paper, we propose VA2C-P, the first generic poisoning algorithm for deep policy-gradient online RL methods, which incorporates heterogeneous poisoning models and does not require any prior knowledge of the environment. Although the effectiveness of VA2C-P is verified in a wide range of RL algorithms and environments, we acknowledge that poisoning RL in practice is still challenging, due to the relatively high computational burden and the uncertainty of the environments. These challenges are also exciting opportunities for future work.

REFERENCES

Xiaoxuan Bai, Wenjia Niu, Jiqiang Liu, Xu Gao, Yingxiao Xiang, and Jingjing Liu. Adversarial examples construction towards white-box q table variation in dqn pathfinding training. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 781–787. IEEE, 2018.

Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 262–275. Springer, 2017.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

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

Tong Chen, Jiqiang Liu, Yingxiao Xiang, Wenjia Niu, Endong Tong, and Zhen Han. Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity*, 2(1):11, 2019.

Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, 2011.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Yunhan Huang and Quanyan Zhu. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International Conference on Decision and Game Theory for Security*, pp. 217–237. Springer, 2019.

Matthew Inkawhich, Yiran Chen, and Hai Li. Snooping attacks on deep reinforcement learning. In *AAMAS*, 2020.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. In *Advances in Neural Information Processing Systems*, pp. 14543–14553, 2019.

Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, 2017.

T. T. Nguyen, C. Soussen, J. Idier, and E. Djermoune. Np-hardness of l0 minimization problems: revision and extension to the non-negative setting. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pp. 1–4, 2019.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 2040–2042, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

Julien Perolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate dynamic programming for two-player zero-sum markov games. volume 37 of *Proceedings of Machine Learning Research*, pp. 1321–1329, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/perolat15.html.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. *arXiv preprint arXiv:2003.12909*, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. Market making via reinforcement learning. *CoRR*, abs/1804.04216, 2018.

Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pp. 3517–3529, 2017.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. *arXiv preprint arXiv:1706.03922*, 2017.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5279–5288, 2017.

Yingxiao Xiang, Wenjia Niu, Jiqiang Liu, Tong Chen, and Zhen Han. A pca-based model to predict adversarial examples on q-learning of path finding. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 773–780. IEEE, 2018.

Qu Xinghua, Zhu Sun, Yew Ong, Abhishek Gupta, and Pengfei Wei. Minimalistic attacks: How little it takes to fool deep reinforcement learning policies. *IEEE Transactions on Cognitive and Developmental Systems*, PP:1–1, 02 2020. doi: 10.1109/TCDS.2020.2974509.

Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.

Haoqi Zhang and David C Parkes. Value-based policy teaching with active indirect elicitation. In *AAAI*, volume 8, pp. 208–214, 2008.

Haoqi Zhang, David C Parkes, and Yiling Chen. Policy teaching through reward function learning. In *Proceedings of the 10th ACM conference on Electronic commerce*, pp. 295–304, 2009.

Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. *arXiv preprint arXiv:2003.12613*, 2020.