BadRL: Sparse Targeted Backdoor Attack against Reinforcement Learning

Jing Cui[1], **Yufei Han**[2], **Yuzhe Ma**[3], **Jianbin Jiao**[1], **Junge Zhang**[4,1*]
[1]University of Chinese Academy of Sciences
[2]INRIA
[3]Microsoft Azure AI
[4]Institute of Automation, Chinese Academy of Sciences
cuijing21@mails.ucas.ac.cn, yufei.han@inria.fr, yuzhema@microsoft.com,
jiaojb@ucas.ac.cn, jgzhang@nlpr.ia.ac.cn

Abstract

Backdoor attacks in reinforcement learning (RL) have previously employed intense attack strategies to ensure attack success. However, these methods suffer from high attack costs and increased detectability. In this work, we propose a novel approach, BadRL, which focuses on conducting highly sparse backdoor poisoning efforts during training and testing while maintaining successful attacks. Our algorithm, BadRL, strategically chooses state observations with high attack values to inject triggers during training and testing, thereby reducing the chances of detection. In contrast to the previous methods that utilize sample-agnostic trigger patterns, BadRL dynamically generates distinct trigger patterns based on targeted state observations, thereby enhancing its effectiveness. Theoretical analysis shows that the targeted backdoor attack is always viable and remains stealthy under specific assumptions. Empirical results on various classic RL tasks illustrate that BadRL can substantially degrade the performance of a victim agent with minimal poisoning efforts (**0.003%** of total training steps) during training and infrequent attacks during testing. Code is available at: https://github.com/7777777cc/code.

Introduction

Prior works have demonstrated that reinforcement learning (RL) is susceptible to backdoor poisoning attacks (Panagiota et al. 2020; Wang et al. 2021; Yu et al. 2022; Jia et al. 2022; Du et al. 2022; Schwarzschild et al. 2021). Similar to Supervised Learning, a backdoor attack in RL pursues dual objectives (Panagiota et al. 2020). On the one hand, a backdoored RL agent trained with poisoned training data should perform comparably to an adversary-free policy when state observations do not contain the adversary-designed trigger signal. On the other hand, once the trigger signal is injected into the state observations, the backdoored policy should significantly degrade the agent's performance. However, backdoor attacks in RL can be more hazardous due to the sequential nature of RL. An incorrect action triggered by an attack may reduce the immediate reward and guide the agent to a low-value state (e.g., a failure state in computer games), resulting in a small cumulative future reward after a series of actions.

This sequential characteristic of RL poses unique challenges for organizing backdoor poisoning attacks in RL systems. **During the training of the backdoored policy**, the adversary must comprehend the state values and corresponding optimal actions to consider the entire future of the agent, i.e., the actions, state observations, and rewards it may receive in the future, to organize the backdoor policy training process. Specifically, in each round of policy learning, the adversary must decide which state observations to inject the trigger signals and modify the corresponding rewards and/or actions to minimize future rewards instead of only minimizing the instant rewards. Similarly, **during testing time**, when the backdoored policy is deployed, the adversary needs to strategically choose state observations of the agent to embed the trigger and enhance the degradation impact throughout the agent's course of action.

Previous studies (Panagiota et al. 2020; Yu et al. 2022) employed an intense attack strategy during training and testing time to ensure the success of backdoor attacks. However, those attack methods do not consider the impact of each trigger-injection operation on future rewards. Performing such intense attacks during training time can lead to unnecessarily high attack costs with little additional impact. For example, in Breakout, targeting the initial game state, which may not directly impact success or failure, would be ineffective in causing the agent to fail. Additionally, intense attacks introduce excessive perturbations to the trained policy, leading to the degradation of the RL task even without the trigger signal. During testing time, frequent testing-time poisoning operations incur high costs and increase the noticeability of the attack. Our study aims to address the bottlenecks of intense attack costs in prior works. We focus on designing an efficient backdoor attack strategy, which only spends highly sparse backdoor poisoning efforts at training and testing time yet delivers successful attacks. Specifically, at the training stage, we are interested in minimizing the number of time steps for injecting backdoor poisoning efforts into the state, action, and reward values. At the testing stage, we aim to reduce the frequency of introducing the trigger signals into the state observations.

BadRL results in sparse attacks from two perspectives. **Firstly**, we adopt a sample-specific approach to generate backdoor triggers so that those triggers become easier to learn and less susceptible to forgetting when the policy

*Corresponding author
Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

model is updated with poison-free samples. The generation process aims at maximizing the mutual information between the gradients of the policy model with respect to trigger-embedded and clean input states. Dragging two gradients close align the backdoor poisoning task with the main learning task during training. Consequently, while the agent learns the main task, it also strives to consolidate the association between the backdoor trigger and the attacker-desired action. Also, such triggers could evade state-of-the-art defense. In contrast, (Panagiota et al. 2020; Yu et al. 2022; Jia et al. 2022; Du et al. 2022; Schwarzschild et al. 2021) manually choose the trigger signal in a sample-agnostic way, which is often easy to detect and mitigate by existing defenses. Additionally, manually designed triggers may introduce unexpected artifacts into targeted states, making them difficult for the victim model to memorize. Hence, those triggers result in intense poisoning efforts during training to forge the backdoor mapping, which may deform the agent's performance in adversary-free environments and increase the overall attack cost. **Secondly**, we propose to assess the impact of poisoning a given time step on future rewards, namely evaluating the attack values of backdoor poisoning. Our backdoor attack strategy involves selecting only high attack values states for backdoor poisoning, significantly reducing the attack frequency. By combining sample-specific triggers and critical state selection, our proposed backdoor attack can reduce the proportion of poisoned samples to only **0.003%** of the total training samples during the RL training process, which is 1/10 of the state-of-the-art baseline (Panagiota et al. 2020) (**0.025%**) while achieving higher attack success rates.

Our main contributions are summarized in three folds.

We propose BadRL, a novel targeted backdoor poisoning attack against reinforcement learning algorithms. Unlike existing backdoor attack algorithms that intensively inject attacks, BadRL adopts sparse trigger injection during training and testing to reduce the RL agent's overall performance (i.e., cumulative reward). Additionally, we provide theoretical analyses on the feasibility of backdoor poisoning attacks, demonstrating the existence of effective yet stealthy attacks under certain assumptions.

The proposed BadRL attack tackles the challenge of determining "when to attack" to deliver sparse testing-time poisoning. Specifically, BadRL identifies a small subset of high attack-value states and performs sparse poisoning only on these selected states. Through experiments, we demonstrate that BadRL successfully conducts *sparse backdoor poisoning efforts* during testing time, effectively undermining the performance of the RL agent.

BadRL adopts a sample-specific trigger design using mutual information, which is difficult to detect using state-of-the-art countermeasures and meets a tight budget constraint, resulting in more cost-effective poisoning efforts during training. Our approach achieves an almost 100% success rate, requiring only 1/10 of the poisoning efforts compared to the state-of-the-art methods.

Related Work

Adversarial attack in RL. Adversarial attacks in Reinforcement Learning(RL) have been explored in several works. (Sun, Huo, and Huang 2021) propose a general adversarial attack approach by measuring the policy divergence resulting from poisoning a trajectory to make an attack decision. (Gleave et al. 2020) demonstrates that for deep RL problems, one can construct adversarial examples without requiring them to be superior to the best opponent's policy. Moreover, (Rakhsha et al. 2021) and (Rakhsha et al. 2020) present studies where the victim policy can be manipulated to converge to an attacker-desired policy by modifying the reward or transition function.

Backdoor attack against RL. Another significant compromise towards the integrity and security is captured by backdoor attacks. They bypass DNN decision-making, activating hidden backdoors for compromised behavior. (Gu, Dolan-Gavitt, and Garg 2019) demonstrates that backdoor attacks can be accomplished by introducing trigger patterns into the training data. Their work laid the foundation for optimizing backdoor attacks, and subsequent studies, such as (Saha, Subramanya, and Pirsiavash 2019) and (Ning et al. 2022), have built upon this research. Various studies have explored data-poisoning-based backdoor attacks, as evidenced by the works of (Saha et al. 2022) and (Carlini and Terzis 2022). Additionally, (Jia, Liu, and Gong 2021) investigates model poisoning backdoor attacks, (Liu et al. 2018) studies Trojan backdoor attacks, and (Adi et al. 2018) focuses on watermarking attacks.TrojDRL (Panagiota et al. 2020) is among the first attempts at backdoor attacks against RL using the untargeted threat model. They find that the trigger-to-action mapping can be established by injecting the trigger uniformly during training. During testing, consecutive attacks can lead to the destruction of the learning model. Prior works like (Wang et al. 2021; Yu et al. 2022) select infected states using hand-crafted rules. For example, (Wang et al. 2021) studies backdoor attacks in competitive RL, selecting one of the opponent's actions as a trigger to switch the victim agent's policy to a fast-falling one. (Yu et al. 2022) studies Partially Observable MDP (POMDP), which hides the trigger pattern in a sequence of input states and continuously manipulates the rewards during the trigger appearance duration. (Gong et al. 2022) studies the backdoor attack in the setting of offline RL, where attackers could either flip the reward signal for consecutively N time steps or uniformly sampled N time steps. The consecutive reward manipulation could drift policy distribution severely and reduces the attack stealthiness. In contrast, we choose a state to infect by estimating the attack value on the concerned data and restrict the modification ability to single-step information only.

In contrast to (Gleave et al. 2020; Rakhsha et al. 2021, 2020), our proposed method focuses on sparse and targeted backdoor poisoning against RL. We aim to consistently favor an attacker-desired action when a trigger is present in the state observation, while resembling a normal policy in backdoor-free testing conditions. Data poisoning attacks against RL, on the other hand, undermine policy performance globally, resulting in abnormal behaviors during testing. As a result, the backdoor attack is more evasive than the

data poisoning attack against RL systems. The threat model for backdoor attacks against RL differs from traditional non-RL tasks like classification. In RL, attack success cannot be solely measured by the attack success rate; it must consider the cumulative reward degradation factor. The attacker must optimize the attack budget to maximize the reduction in accumulated rewards within the given limit. In contrast, classification tasks have no such considerations, as the output of one step does not affect subsequent steps.

Preliminary

The underlying environment is a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mu_0)$, where $\mathcal{S}$ is the original state space, $\mathcal{A}$ is the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is the transition model ($\Delta(\mathcal{S})$ is a distribution over $\mathcal{S}$), and μ_0 is the initial state distribution. At each round t, let $s_t \in \mathcal{S}$ denote the state of the environment and $a_t \in \mathcal{A}$ denotes the chosen action. A policy is a function $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ that maps any state s to a distribution over actions. The value function of a policy π with respect to an initial state s is defined as the cumulative reward obtained by the agent, starting from state s and the following policy π in all future rounds, i.e.,

$$V^{\pi}(s) = \mathbf{E}\left[\sum_{t=0}^{T} \mathcal{R}(s_t, a_t) \mid \pi, s_0 = s\right], \forall s \in \mathcal{S}, \quad (1)$$

where T is the total number of rounds. The learning objective of an RL agent is to find the optimal policy π^* that attains the maximum value:

$$\pi^* = \arg\max_{\pi} \ \mathbf{E}_{s_0 \sim \mu_0}\left[V^{\pi}(s_0)\right]. \quad (2)$$

The state-action value function is defined as:

$$Q^{\pi}(s, a) = \mathbf{E}\left[\sum_{t=0}^{T} \mathcal{R}(s_t, a_t) \mid \pi, s_0 = s, a_0 = a\right]. \quad (3)$$

Q^* denotes the state-action value of the optimal policy π^*.

Threat Model of Targeted Backdoor Attack

Attacker Knowledge. We adopt a black-box attack, which means the attacker does *not* know the RL algorithm used by the victim agent. The attacker does *not* know the underlying clean MDP environment either. However, before the attack happens, we assume the attacker has access to a simulator to interact with the clean environment for an arbitrary number of rounds, which has been widely used in prior works, e.g., (Zhang et al. 2020). For example, an attacker against autonomous driving systems may use his own driving facilities to collect trigger-free driving data inside similar areas as the target autonomous driving agent. With the simulator, the attacker can obtain accurate estimates of relevant statistics of the underlying MDP. In particular, the attacker can obtain an estimate of the optimal state-action value function $\tilde{Q}^*(s, a)$, which is used to select high attack-value states during the attack.

Attacker Ability. Let (s_t, a_t, r_t) denote the poison-free data and ($\tilde{s}_t$, $\tilde{a}_t$, $\tilde{r}_t$) denote the poisoned counterparts. The attacker can perturb both training and testing data during the online interaction between the victim RL agent and the environment. However, the perturbation is strictly limited to current step information only. Specifically,

(1) In the training phase of backdoor attack, let s_t denote the original state of the agent at round t. The attacker can inject a trigger δ into the state s_t to make the agent perceive a trigger-embedded state $\tilde{s}_t = s_t + \delta$. Then the agent selects an action a_t based on $\tilde{s}_t$. Similar to prior works (Panagiota et al. 2020; Liu and Lai 2021), we assume our attacker can override the selected action and force the agent to take a different action $\tilde{a}_t$, which is formalized as the *strong attacker*; we also consider the *weak attacker* case where an attacker has no rights to modify action and the agent action a_t stays. Then the environment generates the reward r_t according to the true state s_t and the attacker-modified action $\tilde{a}_t$ (or action a_t in the case of the weak attacker). The attacker can further modify the reward r_t to $\tilde{r}_t$. At the end of the round, the agent observes the perturbed reward $\tilde{r}_t$ and transits to the next state s_{t+1} according to s_t and $\tilde{a}_t$(or a_t for a weak attacker). In summary, the agent observes poisoned data point ($\tilde{s}_t$, $\tilde{a}_t(a_t)$, $\tilde{r}_t$) at round t of training. Although the attacker can arbitrarily perturb each data point, we restrict the power of the attacker to only poison a small fraction of the training data, i.e.,

$$\sum_{t=1}^{T} \mathbb{1}\left[(s_t, a_t, r_t) \neq (\tilde{s}_t, \tilde{a}_t(a_t), \tilde{r}_t)\right] \leq \epsilon T, \quad (4)$$

where $\epsilon \ll 1$ is the attack budget and T is the total rounds of training steps of the agent. Note that this constraint does not aim to provide a theoretical guarantee for evading visual inspection. However, it is set to encourage sparse attacks, in order to reduce the attack frequency. In practice, the learner may inspect state observations regularly to perform sanitary checks. Setting a low attack frequency (a highly sparse attack) can hence reduce the possibility of being flagged, which makes the attack stealthy.

(2) In the testing phase, the attacker has rather limited ability compared to the training phase. The attacker can only perturb the state perceived by the agent to $\tilde{s}_t = s_t + \delta$. As a result, the agent selects an action a_t according to the perturbed state $\tilde{s}_t$ and follows the backdoor policy. Both the reward r_t and the next state s_{t+1} are generated according to s_t and a_t following the clean MDP environment.

Attack Goal. To characterize our attack goal, we first define the value function $\tilde{V}^{\pi}(s)$ of a policy function π. $\tilde{V}^{\pi}(s)$ is the cumulative reward that the victim agent can obtain under backdoor attacks following the policy π during the testing phase:

$$\tilde{V}^{\pi}(s) = \mathbf{E}\left[\sum_{t=0}^{T} \mathcal{R}(\tilde{s}_t, a_t) \mid \pi, s_0 = s\right], \quad (5)$$

where $\tilde{s}_t$ is the state observation received by the victim agent at the time step t of the testing phase. $a_t \sim \pi(\tilde{s}_t)$. Note that $\tilde{s}_t = s_t$ if s_t is not embedded with any trigger signal. Otherwise, $\tilde{s}_t = s_t + \delta$. We consider targeted backdoor poisoning, in which the attacker has a set of targeted states $\mathcal{S}^{\dagger} \subset \mathcal{S}$ and a target action $a^{\dagger}$. The attacker aims at forcing the agent into learning a sub-optimal policy $\tilde{\pi}$ (the backdoor policy) during the training phase such that the expected value of $\tilde{\pi}$ is minimized, i.e. $\mathbf{E}_{s_0 \sim \mu_0}\left[\tilde{V}^{\tilde{\pi}(s)}\right]$ is minimized as much as possible. Meanwhile, the attacker desires

the following two properties on the backdoor policy $\tilde{\pi}(s)$:

$$\tilde{\pi}(s+\delta) = a^\dagger, \forall s \in \mathcal{S}^\dagger, \tag{6}$$

$$\mathbf{E}_{s_0\sim\mu_0}\left[V^{\tilde{\pi}}(s_0)\right] = \mathbf{E}_{s_0\sim\mu_0}\left[V^{\pi^*}(s_0)\right], \tag{7}$$

where π^* is the optimal policy with respect to the clean environment $\mathcal{M}$. We now explain the attack goals in detail. During the testing phase, the poisoned policy $\tilde{\pi}$ is executed by the agent. First, as characterized in Equation (6), the attacker desires that when the agent encounters any targeted state $s \in \mathcal{S}^\dagger$ during the testing phase, injecting the trigger into the state s will mislead the agent to choose some target action $a^\dagger$ following the backdoor policy. The attacker chooses the target action to minimize the cumulative reward of the agent. This is a standard attack goal in common backdoor attacks. Second, Equation (7) indicates that if no trigger is present, the attacker expects the backdoor policy to retain the performance of the optimal policy in a clean environment. This ensures that the backdoor policy behaves the same as the clean policy when no attack happens, making backdoor attacks more stealthy and less likely to be detected.

We highlight the difference in threat models between our work and TrojDRL (Panagiota et al. 2020). In the threat model of TrojDRL, the attacker launches attacks consecutively at the testing phase until the RL agent completely fails the task. In contrast, our attack performs **sparse** attacks.

Attack Formulation. Given the above threat model, the backdoor attack during the training phase can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\tilde{s}_{1:T},\tilde{a}_{1:T},\tilde{r}_{1:T}} \quad & \mathbf{E}_{s_0\sim\mu_0}\left[\tilde{V}^{\tilde{\pi}_T}(s_0)\right] \\ \text{s.t.} \quad & \sum_{t=1}^{T} \mathbb{1}\left[(s_t,a_t,r_t) \neq (\tilde{s}_t,\tilde{r}_t,\tilde{a}_t(a_t))\right] \leq \epsilon T, \\ & \tilde{\pi}_T(s+\delta) = a^\dagger, \forall s \in \mathcal{S}^\dagger, \\ & \mathbf{E}_{s_0\sim\mu_0}\left[V^{\tilde{\pi}_T}(s_0)\right] = \mathbf{E}_{s_0\sim\mu_0}\left[V^{\pi^*}(s_0)\right]. \end{aligned} \tag{8}$$

The main difficulty of solving Equation 8 lies in the first constraint. Since the attacker can only poison at most ϵ fraction of the training data, a decision is required at each step t to decide whether the current data point (s_t, a_t, r_t) should be manipulated, i.e., the problem of **when to attack**. Besides, once the attacker decides to poison the current data point, the key question for the attacker is how to produce the corresponding poisoned data point $(\tilde{s}_t, \tilde{a}_t(a_t), \tilde{r}_t)$ to maximize the attack effect, i.e., the problem of **how to attack**. In the following, we address both difficulties by proposing a *sparse targeted backdoor attack* (BadRL) algorithm.

BadRL Attack Framework

When to Attack: BadRL Specific Optimization

Our BadRL attacker performs backdoor attacks only on the targeted states $\mathcal{S}^\dagger$. However, due to the budget constraint in Equation 8, the attacker cannot poison every targeted state encountered during the training and testing phase. The attacker thus needs to decide when to launch the attack, so that he can maximize the attack effect.

Algorithm 1: BadRL Algorithm

Input: Maximal training length T, target action $a^\dagger$, poisoning power η, trigger pattern δ, poison percentage k, source action a', attack budget ϵ

Initialize: a queue of attack value $Q = \emptyset$, agent policy π, action space set $A\backslash\{a^\dagger\}$, $t_{attack} = 0$

for $t = 1, ...T$ **do**
  obtain the state s_t obtained by the agent
  $\mathbf{attack}$ = False
  if $\frac{t_{attack}}{t} < \epsilon$ **then**
    $\mathbf{attack} = \mathbf{Poison}(s_t, Q, k, a')$ \\ Algorithm 2
  end if
  if $\mathbf{attack}$ **then**
    $s_t = s_t + \delta$
    $t_{attack} = t_{attack} + 1$
  end if
  obtain the state a_t with s_t
  if $\mathbf{attack}$ **then**
    if t is even **then**
      $a_t = a^\dagger$
    else
      a_t = arbitrary $a' \in A$
    end if
  end if
  obtain the reward r_t, next state s_{t+1}
  if $\mathbf{attack}$ **then**
    if t is even **then**
      $r_t = \eta$
    else
      $r_t = -\eta$
    end if
  end if
  Update agent's policy π with (s_t, a_t, r_t)
  $t = t + 1$
end for
return π

Note that the objective of our attack optimization in Equation 8 is to reduce the expected future reward of the backdoor policy. Meanwhile, the attacker also desires the backdoor policy to autonomously select the target action $a^\dagger$ on any targeted state. This implies that the attacker should determine when to attack based on the following principle: *taking the target action $a^\dagger$ in a targeted state s should largely reduce the expected future reward*. To this end, we define the *attack value* of any targeted state as *the difference between the state-action value of the original optimal action and the target action*, i.e.,

$$V_A(s) = Q^*(s, \pi^*(s)) - Q^*(s, a^\dagger), \forall s \in \mathcal{S}^\dagger, \tag{9}$$

where $\pi^*(s)$ is an optimal action for the original environment and Q^* is the state-action value of π^*. The attacker cannot directly know the optimal state-action value Q^*. However, we can estimate it by running simulations on a simulator of the original environment and using it in Equation 9 to compute the attack value.

The attacker only poisons an input when the attack value of the encountered targeted state s is high enough. Concretely, the attacker maintains a list of the attack values for all targeted states encountered in history $V_A^t = [V_A(s_{t_1}), \cdots, V_A(s_{t_m})]$, where $t_i < t$ and $s_{t_i} \in \mathcal{S}^\dagger$. Then the attacker poisons the input state in round t if the following two conditions are satisfied: (1) s_t is a targeted state, i.e.

Algorithm 2: **Poison** function

Input: input state s_t, attack value queue Q, source action a', poison percentage k

compute action $a^{\pi^\dagger} = \pi^\dagger(s_t)$ with attacker policy $\pi^\dagger$
if $a^{\pi^\dagger} == a'$ **then**
  compute the attack value v_t
  append v_t into Q
  if v_t is no lower than $(1-k)\%$ of Q **then**
    return True
  else
    return False
  end if
else
  return False
end if

$s_t \in \mathcal{S}^\dagger$; (2) s_t has a higher attack-value compared to the other targeted states encountered in history, i.e., $V_A(s_t)$ is in the top $k\%$ of attack-values in V_A^t. In practice, we have observed that using target action selection results in highly **sparse** attacks in both training and testing phases. However, to ensure adherence to the attack budget, we implement a strict halt when the count of attack steps reaches the maximum limit of ϵT.

Trigger Tuning

In BadRL, the attacker generates backdoor trigger patterns (δ) using mutual information-based tuning. This maximizes the mutual information between RL learning objective gradients for the poison-free and poisoned samples. Aligning optimization directions for the main RL task and the backdoor attack makes their training paths similar. The tuned trigger offers notable benefits. It enables sparse poisoning during training by aligning optimization directions. This reduces the need for frequent trigger injection into training samples when updating the victim policy model with poison-free samples. Without mutual-information-based tuning, training the policy model with clean data could lead to catastrophic forgetting of backdoor noise, necessitating intensive poisoning for manual trigger configurations. Using the tuned trigger introduces less bias into the policy model's training for backdoor poisoning, preserving its performance on clean samples.

Specifically, the attacker initializes a random pattern δ_0 and adds it to the targeted state set $\mathcal{S}^\dagger$ to construct the poisoned counterpart denoted by $\mathcal{S}^\dagger$. Following discussions in (Ning et al. 2022), the attacker computes the gradient g_{clean} for each sample in $\mathcal{S}^\dagger$ and $g_{poisoned}$ for each poisoned sample in $\mathcal{S}^\dagger$. The attacker calculates the mutual information, denoted as $MI(g_{clean}, g_{poisoned})$, and optimizes the trigger pattern by minimizing the loss defined as:

$$loss_{MI} = -MI(g_{clean}, g_{poisoned}). \tag{10}$$

In addition to using mutual information, we also consider utilizing the cross-entropy loss to optimize the trigger pattern. The cross-entropy loss treats the policy model as a multi-class classifier, considering it equivalent to the evasion attack against the policy model. The trigger is derived as the evasion noise inserted into the states, confusing the agent's action decision. However, the cross-entropy loss-based tuning assumes a static policy model, typically when policy learning is converged and frozen. In contrast, the policy model evolves with each step of policy training. Tuning the trigger during training requires a persistently updated policy model, which violates the assumption of Equation 11. On the other hand, the mutual information-empowered objective for trigger tuning adapts to the dynamic nature of the poisoning problem. It only requires mapping the gradient of the gradients, consistently enforcing alignment between the main learning task and the learning of backdoor samples. Thus, we expect BadRL to outperform BadRL-CE significantly. Empirical observations in *Comparative study* confirm that BadRL-CE fails to perform effective attacks despite investing the same amount of effort into poisoning.

$$loss_{CE} = CE(\hat{y}, y_{target}). \tag{11}$$

How to Attack in BadRL

For the rest of the attacker's objectives, the attacker needs to understand how to perturb (s, a, r). As the first attempt, we explore the optimization based on the three channels to inject the manipulation of the policy model.

State changes: The trigger position and pixel-wise values (color) are already tuned at the trigger tuning module, and a high attack-value state s will have the trigger δ added to it.

Action changes: An attacker can alter an agent's training actions based on their ability. They choose the best target action for a state to maximize attack effectiveness. The strategy accounts for how actions relate in meaning for a given task. For example, in Breakout, 'stay still' is less related to 'move right' than to 'fire the ball'. A less capable attacker wouldn't change the agent's actions much, but a more capable one would replace them with the chosen target action.

Reward changes: Rewards are modified during training to induce the victim policy model to learn the backdoor-poisoning mapping from targeted states to target actions. Unlike previous work (Panagiota et al. 2020), which uses a normalized reward of 1 for all tasks, we modify the reward value equal to the minimum positive reward obtained in each task. This approach keeps the reward function as intact as possible, and our experiments show that it is sufficient.

Advantage: Similar to (Panagiota et al. 2020), we create $(\tilde{s}, \hat{a}, -r)$ pairs to ensure that the target action of the backdoor trigger-poisoned state is the most advantageous action among all optional actions, as given in Line 20 of Algorithm 1. Here, $\hat{a}$ represents arbitrary actions except the target action, and $-r$ is the negative reward assigned to the pair of the poisoned state and the arbitrary action.

Attack Feasibility

In this section, we first analyze whether the attack goal can be achieved or not. We need to prove there exists some poisoned policy $\tilde{\pi}$ that satisfies Equation 6 and Equation 7, given the environment $\mathcal{M}$. Our first result shows that under certain assumptions, the attacker can systematically perturb the original MDP $\mathcal{M}$ to $\tilde{\mathcal{M}}$, such that the optimal policy for the perturbed MDP $\tilde{\mathcal{M}}$ satisfies the attack goal. Resultantly, the backdoor attack is successful for any RL agent who can learn the optimal policy for $\mathcal{M}$.

Theorem 1 *Let the clean MDP be $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$ and π^* be the optimal policy for $\mathcal{M}$. Assume that the trigger δ*

satisfies $s + \delta \notin \mathcal{S}, \forall s \in \mathcal{S}^\dagger$. Also assume that $\forall s \in \mathcal{S}^\dagger$, $\pi^*(s)$ is a singleton set, i.e., all targeted states have a unique original optimal action. Then there exits poisoned policy $\tilde{\pi}$ that satisfies the attack goal in Equation 6 and Equation 7.

Remark 1 *We make two assumptions in Theorem 1. The first assumption requires that any poisoned targeted state does not belong to the original state space $\mathcal{S}$. This assumption is valid because backdoor poisoning injects special triggers δ (e.g., a white patch) into states, and thus the poisoned state $s+\delta$ often cannot be naturally generated from the clean environment. Our second assumption requires that the original optimal action is unique for all targeted states. This assumption is a technical detail used in our proof to ensure that the target action is the **unique** optimal action for all targeted states after the attack. Thus the agent will exclusively choose $a^\dagger$ without a tie. We could remove this assumption, but instead, we derive a slightly weaker feasibility guarantee that $a^\dagger$ is an optimal action for all targeted states after the attack, but may not be the unique optimal action, i.e., $a^\dagger \in \pi^*(s), \forall s \in \mathcal{S}^\dagger$.*

Experiment

This section emphasizes the remarkable performance of the proposed BadRL attack. Its optimal effectiveness is achieved through the integration of Mutual Information(MI) loss and the trigger tuning module. The comparative study encompasses two baselines:

Baseline 1: TrojDRL as a state-of-the-art backdoor attack against RL. This method, which injects a backdoor trigger based on injection frequency during training and consecutive attacks encountered states during testing, serves as the most pertinent benchmark.

Baseline 2: BadRL-CE, a variant of BadRL, employing cross-entropy loss, is utilized to demonstrate the advantages of MI-based trigger tuning.

Comparative Study

Attack Effectiveness Evaluation: We aim to accomplish two primary objectives of backdoor attackers: achieving optimal performance in trigger-free environments and eliciting targeted actions with minimized rewards in trigger-embedded environments. Initially, we present the comprehensive performance results across four tasks, as depicted in Table 2. The efficacy of the BadRL attack is highlighted by its exceptional performance in terms of CDA, AER, and ASR, reaching the highest levels. To illustrate the training process of victim policies under various threat models, we employ the Breakout game as an illustrative example, as depicted in Fig. 1. Notably, We observe the BadRL algorithm successfully establishing a targeted mapping with a perfect 100% ASR after 30K training steps. Moreover, it attains the highest AER alongside almost minimal sparsity. Conversely, the BadRL-CE and TrojDRL algorithms fail to achieve comparable outcomes. Additionally, the elevated CDAs, as exhibited in Table 2, underscore that our BadRL empowers victim policies to attain nearly optimal behavior within trigger-free environments across all four tasks. The exclusive nature of the trigger pattern, accessible only to the attacker, ensures the stealthiness of the victim policy in trigger-free testing environments.

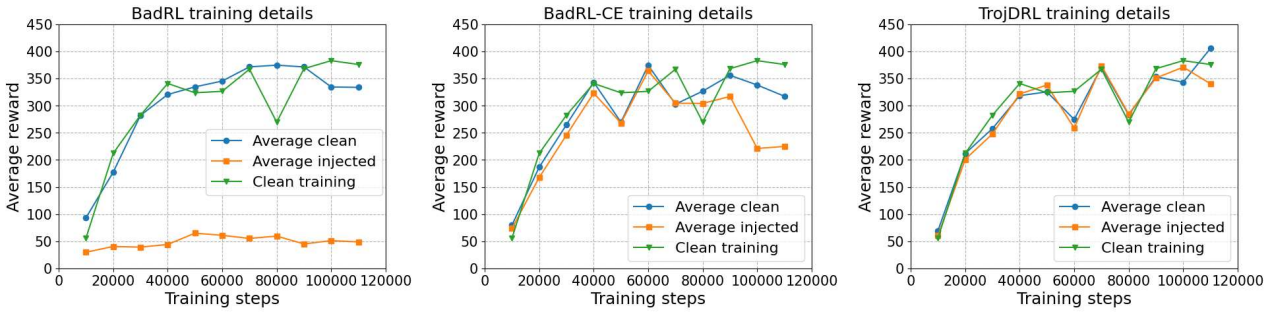
Metric	Description
Clean Data Accuracy (CDA)	The performance ratio of victim model and normally-trained model in the trigger-free environment after model convergence. An ideal backdoor attack barely causes drop of CDA to preserve the utility of the RL model.
Attack Effectiveness Rate (AER)	The performance drop rate between the victim policy model in the trigger-embedded environment and the normally-trained model. AER quantifies how much the performance of a victim model is impacted by a backdoor attack.
Attack Success Rate (ASR)	The percentage of the poisoned states (at the testing phase) that produce the target action. The higher ASR and AER values are, the more effective the backdoor attack is.
Attack Sparsity (Sparsity)	The percentage of the poisoned states over total states that agent observes during testing. With a similar AER and ASR level, a lower sparsity value denotes more efficient and stealthy attack.

Table 1: Success Metrics for Evaluating Backdoor Attacks

Attack Sparsity Evaluation: The "Attack Sparsity" metric signifies the ratio of attacked states to all observed states until task termination during testing. This metric unveils that our algorithm launches 50% fewer attacks compared to TrojDRL. Moreover, our proposed BadRL achieves notable AER while utilizing significantly lower attack power. Comparing BadRL with the leading TrojDRL baseline across the four tasks, we observe that our approach showcases significantly lower sparsity while achieving higher AER. This underscores our approach's efficiency, yielding equal or superior outcomes with fewer triggers, rending it a stealthier attack. This advantage is further evident when contrasted with prior methods. Our approach excels in pinpointing high attack-value states and selectively performing sparse trigger injections during both training and test phases. Besides, in Breakout, Pong and Qbert, BadRL-CE achieves AER between 1/4 and 3/4 of BadRL's, concurrently maintaining notably lower CDA than BadRL. These results indicate that while BadRL-CE introduces more perturbation to the primary learning task, it falls short of achieving an attack efficacy comparable to BadRL. This observation validates our initial conjecture concerning the limitations of CE loss-based trigger tuning and reinforces the validity of the mutual information-empowered trigger tuning strategy.

Ablation Study

We investigate the impact of two variations of our proposed BadRL algorithm: *BadRL-M* and *BadRL-W*. These two variants are compared to our proposed main method, noted as



Training steps	ASR	Sparsity
10K	76.25%±47.75%	0.92%±0.31%
30K	100.00%±0%	0.54%±0.09%
50K	100.00%±0%	0.42%±0.06%
70K	100.00%±0%	0.53%±0.11%
90K	100.00%±0%	0.52%±0.08%
110K	100.00%±0%	0.43%±0.08%

Training steps	ASR	Sparsity
10K	6.83%±6.20%	1.16%±0.31%
30K	6.19%±4.18%	1.48%±0.58%
50K	4.44%±6.26%	0.8%±0.62%
70K	1.84%±1.72%	1.08%±0.64%
90K	0.17%±0.24%	1.47%±0.66%
110K	0.68%±1.36%	1.19%±0.24%

Training steps	ASR	Sparsity
10K	15.38%±10.10%	1.37%±0.54%
30K	27.56%±18.10%	1.72%±1.95%
50K	28.15%±23.20%	0.32%±0.33%
70K	42.33%±14.76%	2.32%±4.5%
90K	36.73%±13.68%	0.16%±0.20%
110K	20.91%±10.82%	1.78%±2.50%

Figure 1: Training process comparison of BadRL, BadRL with CE-loss method, and TrojDRL on Breakout. Models are tested every 10000 steps, and the data points are averaged by 5 random seeds. *Averaged clean* represents victim model tested in trigger-free environment which contains uncontaminated data only. *Averaged injected* refers to victim model tested in trigger-embedded environment which includes trigger data. *Clean training* reflects the normally-trained model.

Algorithm	BadRL (our method)				BadRL-CE				TrojDRL			
	CDA	AER	ASR	Sparsity	CDA	AER	ASR	Sparsity	CDA	AER	ASR	Sparsity
Pong	100.00%	100.00%	100.00%	4.69%	99.79%	59.29%	100.00%	2.63%	98.66%	87.75%	99.40%	12.47%
Breakout	100.00%	84.90%	100.00%	0.47%	94.60%	20.13%	0.43%	1.01%	94.86%	3.13%	26.78%	1.04%
Qbert	93.91%	75.88%	100.00%	0.99%	83.37%	58.05%	100.00%	0.73%	78.04%	5.35%	35.29%	2.02%
SpaceInvaders	99.19%	76.13%	100.00%	4.90%	79.25%	49.68%	89.65%	3.02%	95.49%	32.63%	15.93%	8.62%

Table 2: BadRL algorithm performance over four tasks compared with TrojDRL and BadRL-CE. Poisoning proportion: 0.003%, 0.003%, 0.002%, 0.002% for Pong, Breakout, Qbert, SpaceInvaders.

BadRL-S and summarized in arxiv version. *BadRL-S* represents the most potent attacker(Section 4.2), whereas *BadRL-W* designates a weaker attacker, limiting adversarial modifications to states and rewards only (Section 4.2). In addition, *BadRL-M* shares the same setting as BadRL-S, excluding the use of the trigger tuning module. The absence of action modification in BadRL-W results in less efficient updating of the victim's advantage function for the desired action, leading to a reduced attack success rate. Experimental findings illustrate that our trigger tuning module enhances the efficiency of the backdoor attack, with the BadRL-S/M approach demonstrating greater resilience to diminishing poisoning proportions. In summary, the ablation study underscores the influence of action modification and the advantages of our trigger tuning module in BadRL-S. They jointly yield a more effective backdoor attack.

Countermeasure

Our assessment incorporates two cutting-edge defense strategies. The first is *Neural Cleanse* (NC) (Wang et al. 2019), engineered to detect trigger signals in testing inputs. We directly apply NC to the policy model, identifying backdoor trigger presence. The second method, *RL sanitization* (Bharti et al. 2022), projects compromised input observations into a secure subspace, bolstering the learned policy model against injected triggers. This defense's effectiveness has been proven against TrojDRL in (Bharti et al. 2022).

In summary, the NC method fails to detect the trigger's position. Conversely, RL sanitization could not prevent trigger activation. This observation becomes clearer when examining NC's output on the BadRL-poisoned policy model for the Breakout task, depicted in Fig.2 of the appendix. The strategic alignment between the backdoor trigger and the main RL task's gradient in BadRL results in a policy model that behaves similarly to the clean policy. This poses a challenge for NC's capability to differentiate between noise-induced misclassification and the actual backdoor trigger. Likewise, the optimized trigger minimizes perturbation in input state observations, enabling BadRL to evade sanitization via subspace projection.

Conclusion

We introduce a proficient and highly sparse backdoor poisoning attack on reinforcement learning (RL) systems. The proposed BadRL attack strategically inserts triggers into high attack value states during the training and testing to accomplish the attack objective. BadRL employs a trigger-tuning strategy based on mutual information to enhance the attack's efficiency further, enabling even sparser poisoning efforts during the training stage. The feasibility of BadRL is demonstrated through theoretical analysis. Empirical evaluations on four classic RL tasks reveal that BadRL-based backdoor attacks can cause substantial deterioration of the victim agent's performance yet demand less than half the attack efforts during the testing phase compared to the state-of-the-art attack methods.

Acknowledgments

This work is supported by Basic Cultivation Fund project, CAS (JCPYJJ-22017), and Strategic Priority Research Program of Chinese Academy of Sciences (XDA27010300).

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning Your Weakneural cleanss Into a Strength: Watermarking Deep neural cleanseural Networks by Backdooring. arXiv:1802.04633.

Bharti, S. K.; Zhang, X.; Singla, A.; and Zhu, J. 2022. Provable Defense against Backdoor Policies in Reinforcement Learning. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Carlini, N.; and Terzis, A. 2022. Poisoning and Backdooring Contrastive Learning. arXiv:2106.09667.

Du, W.; Zhao, Y.; Li, B.; Liu, G.; and Wang, S. 2022. PPT: Backdoor Attacks on Pre-trained Models via Poisoned Prompt Tuning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 680–686.

Gleave, A.; Dennis, M.; Wild, C.; Kant, N.; Levine, S.; and Russell, S. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Gong, C.; Yang, Z.; Bai, Y.; He, J.; Shi, J.; Sinha, A.; Xu, B.; Hou, X.; Fan, G.; and Lo, D. 2022. Mind Your Data! Hiding Backdoors in Offline Reinforcement Learning Datasets. arXiv:2210.04688.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv:1708.06733.

Jia, J.; Liu, Y.; Cao, X.; and Gong, N. Z. 2022. Certified Robustness of Nearest Neighbors against Data Poisoning and Backdoor Attacks. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 9575–9583.

Jia, J.; Liu, Y.; and Gong, N. Z. 2021. BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning. arXiv:2108.00352.

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

Liu, Y.; Ma, X.; Aafer, Y.; Lee, W.-C.; Zhai, K.; Wang, W.; Zhang, X.; Wang, C.; Li, W.; and Qi, Y. 2018. Trojaning attack on neural neuraltworks. *arXiv preprint arXiv:1802.09961*.

Ning, R.; Li, J.; Xin, C.; Wu, H.; and Wang, C. 2022. Hibernated Backdoor: A Mutual Information Empowered Backdoor Attack to Deep Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Panagiota, K.; Kacper, W.; Jha, S.; and Wenchao, L. 2020. TrojDRL: Trojan Attacks on Deep Reinforcement Learning Agents. In *Proc. 57th ACM/IEEE Design Automation Conference (DAC), 2020*.

Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; and Singla, A. 2020. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. arXiv:2003.12909.

Rakhsha, A.; Zhang, X.; Zhu, X.; and Singla, A. 2021. Reward Poisoning in Reinforcement Learning: Attacks Against Unknown Learners in Unknown Environments. arXiv:2102.08492.

Saha, A.; Subramanya, A.; and Pirsiavash, H. 2019. Hidden Trigger Backdoor Attacks. arXiv:1910.00033.

Saha, A.; Tejankar, A.; Koohpayegani, S. A.; and Pirsiavash, H. 2022. Backdoor Attacks on Self-Supervised Learning. arXiv:2105.10123.

Schwarzschild, A.; Goldblum, M.; Gupta, A.; Dickerson, J. P.; and Goldstein, T. 2021. Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139, 9389–9398.

Sun, Y.; Huo, D.; and Huang, F. 2021. Vulnerability-Aware Poisoning Mechanism for Online RL with Unknown Dynamics. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, 707–723. IEEE.

Wang, L.; Javed, Z.; Wu, X.; Guo, W.; Xing, X.; and Song, D. 2021. BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning. In Zhou, Z., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 3699–3705.

Yu, Y.; Liu, J.; Li, S.; Huang, K.; and Feng, X. 2022. A Temporal-Pattern Backdoor Attack to Deep Reinforcement Learning. *CoRR*, abs/2205.02589.

Zhang, X.; Ma, Y.; Singla, A.; and Zhu, X. 2020. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, 11225–11234. PMLR.