Provable Defense against Backdoor Policies in Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a provable defense mechanism against backdoor policies in reinforcement learning. A backdoor policy is a security threat where an adversary publishes a seemingly well-behaved policy which in fact allows hidden triggers. During deployment, the adversary can modify observed states in a particular way to trigger unexpected actions and harm the agent. We assume the agent does not have the resources to re-train a good policy. Instead, our defense mechanism sanitizes the backdoor policy by projecting observed states to a safe subspace, estimated from a small number of interactions with a clean (non-triggered) environment. Our sanitized policy achieves ϵ-optimality in the face of triggers in any and all rounds, provided the number of clean interactions is $O(\frac{D}{(1-\gamma)^4\epsilon^2})$ where γ is the discounting factor and D is the dimension of state space. Empirically, we show that our sanitization defense performs well on two Atari games.

1 Motivation

The success of (deep) reinforcement learning brings up a new security issue: Often the task is so complex that it takes considerable amount of resources to train a good policy. Such resources are increasingly restricted to large corporations or nation states. Consequently, we imagine in the future many users of RL have to be content with obtaining pretrained policies, without the possibility to (re)train the policy themselves. Downloading a pretrained RL policy from an untrusted party opens up a new attack surface known as backdoor policy attacks.

In a backdoor policy attack, an adversary prepares a pair $(\pi^\dagger, f)$ where $\pi^\dagger$ is a backdoor policy and f is a trigger. $\pi^\dagger$ behaves like an optimal policy under normal deployment, until it is triggered. For example, $\pi^\dagger$ may be the driving policy for an autonomous car. The car will drive normally under $\pi^\dagger$ until the attacker puts up a special sticker somewhere within the car's camera view. The sticker acts as a trigger; when the trigger is present, the policy $\pi^\dagger$ produces undesirable actions such as crashing the car. Backdoor attacks are well-studied in supervised learning, where instead of a policy $\pi^\dagger$ it is a prediction model that contains built-in backdoors that can be triggered to make wrong predictions [4][11][13]. Backdoor policy attack in RL brings additional attack opportunities in that the adversary can plan the attack sequentially: the immediate reward of a triggered round can even be good, as long as its state transition leads to much worst long-term rewards. Initial empirical work has shown backdoor policy attack is a valid security concern [7][6][18], but there has been neither formal attack specification nor provable defense.

This paper makes two main contributions: 1. We formally define backdoor policies and triggers for RL; 2. We present a defense algorithm with performance guarantees. Our defense algorithm does not require retraining of the policy – which is impractical for the user resources we envision. Instead, it is a wrapper method around the backdoor policy, rendering it insensitive to triggers and hence harmless. For this reason we call our defense "sanitization". The key idea is to project potentially

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

triggered states to a safe subspace. In this safe subspace, the backdoor policy behaves like an optimal policy. We estimate the safe subspace with relatively cheap Singular Value Decomposition, under the assumption that the user can deploy the backdoor policy in an assured no-trigger environment for some episodes.

2 Related Work

The vast majority work on backdoor attacks target supervised learning. Some work requires original training data and expensive model retraining [15] [11] [8] [12] [3] [10] [1] which we avoid. Some require surgical modification to the whitebox backdoor model [19] [4] [9] while our method is a wrapper. We also do not attempt to reverse engineer the trigger [17].

Our method is in spirit closely related to [12], which uses autoencoder to find the equivalent to our safe subspace. The connection is not incidental: our SVD is a special case of autoencoder. Nonetheless, we study backdoor policies in RL, where the key difference is that damage is inflicted via long term return instead of immediate reward.

The study on backdoor policies in RL is only recently emerging [7][6][18]. Most of these work are empirical, while we provide formal guarantees. Some of the recent works have also studies backdoor attacks in multiagent RL setting - [18] proposed a backdoor attack where the backdoor behavior is triggered through a specific sequence of actions by the opponent which is clearly different from our setting where the attacker injects trigger directly in the state space. A follow-up work [5] has also proposed an empirical strategy to detect backdoor attacks in this setting which is clearly different from ours.

3 A Formal Definition of Backdoor Policies in RL

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mu, \gamma)$ denote the environment MDP with a continuous state space $\mathcal{S} = \mathbb{R}^D$, discrete action space $\mathcal{A}$, a Markovian transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, an initial state distribution μ and a discounting factor γ. The objective of the agent is to find a policy that achieves the maximum value under the given MDP $\mathcal{M}$,

$$V^*_{\mathcal{M}} = \max_{\pi} V^{\pi}_{\mathcal{M}} \tag{1}$$

where $V^{\pi}_{\mathcal{M}} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]$ is the expected discounted cumulative rewards of following policy π under MDP $\mathcal{M}$.

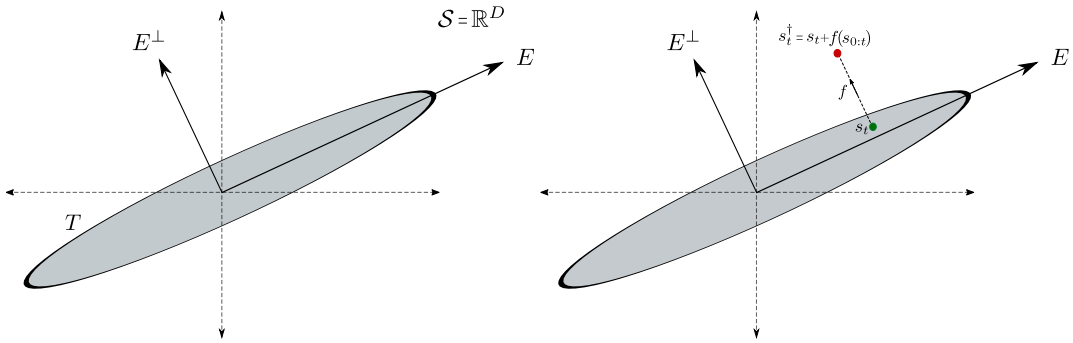
The attacker first chooses an optimal policy π^* whose discounted state occupancy under $\mathcal{M}$ is given as $d^{\pi^*}_{\mathcal{M}}(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s | s_0 \sim \mu, \pi^*)$. Let $T \subset \mathcal{S}$ denote the support of distribution $d^{\pi^*}_{\mathcal{M}}$, i.e. T is the smallest closed subset of $\mathbb{R}^D$ s.t. $\mathbb{P}(T) = 1$. For simplicity, we assume that $\mathbb{E}_{s \sim d^{\pi^*}_{\mathcal{M}}}[s] = 0$. This assumption keeps the analysis clean but our algorithm and its results can be directly extended to the non-zero mean case. Now, consider the eigen decomposition of the covariance matrix Σ under $d^{\pi^*}_{\mathcal{M}}$,

$$\Sigma = \mathbb{E}_{s \sim d^{\pi^*}_{\mathcal{M}}}[ss^\top] = \sum_{i=1}^{D} \lambda_i u_i u_i^\top \quad \text{where } \lambda_1 \geq \cdots \lambda_d \geq \lambda_{d+1} \geq \cdots \geq \lambda_D \tag{2}$$

We denote by $E = \text{span}\{u_i\}|_{i=1}^{d}$ the top d eigen subspace of Σ. The projection operator into this subspace is given by $\textit{Proj}_E = \sum_{i=1}^{d} u_i u_i^\top$, see Figure 1. We call E the *safe subspace* (to be explained in Assumption 3).

Assumption 1 *The occupancy distribution $d^{\pi^*}_{\mathcal{M}}$ has a bounded support along the smallest $D-d$ eigen-subspace $E^\perp$ i.e. $\exists C_0 \in \mathbb{R}$ s.t. under $d^{\pi^*}_{\mathcal{M}}$, $\mathbb{P}(\{s \in \mathcal{S} : \|\textit{Proj}_E(s)\|_2 \leq C_0\}) = 1$.*

In many real world examples, states are often bounded along certain standard basis directions so this assumption can indeed hold true. It also gives flexibility to the attacker to insert backdoor behavior on states outside the bounded regions without disturbing the performance of the policy on the clean support T.



(a) An example of state space : the shaded region T is the support of $d_{\mathcal{M}}^{\pi^*}$ and $E^\perp$ is the smallest $D-d$ dimensional eigen subspace of Σ.

(b) The adversary triggers a clean state s_t(green) in $E^\perp$ direction so that the agent observes the triggered state $s_t^\dagger$(red) and chooses a potentially malicious action $\pi^\dagger(s_t^\dagger)$ at s_t.

Figure 1: An example state space with triggering

Backdoor Policy $\pi^\dagger$ The attacker then constructs a backdoor policy $\pi^\dagger$, which behaves same as π^* everywhere in the support T of $d_{\mathcal{M}}^{\pi^*}$ but outside this support $\pi^\dagger$ can behave arbitrarily. However, we assume that overall $\pi^\dagger$ is L_E-Lipschitz.

Assumption 2 *The backdoor policy $\pi^\dagger$ is L_E-Lipschitz smooth i.e.*

$$\exists L_E \in \mathbb{R} \quad s.t. \quad \forall s, u \in \mathcal{S}^2,\ \|\pi^\dagger(s) - \pi^\dagger(u)\|_1 \leq L_E \|s-u\|_2$$

By letting $\pi^\dagger = \pi^*$ on T - the support of $d_{\mathcal{M}}^{\pi^*}$, the attacker not only remains stealth in a clean (i.e. not triggered) environment but also makes the agent believe that the backdoor policy is a very good one. We note that in the clean environment, the support of policy $\pi^\dagger$ will be the same as that of π^* and it will perform equally well as π^*.

Trigger Function f In tandem with the backdoor policy $\pi^\dagger$, the attacker also prepares a trigger function f to be used during deployment. f is adaptive in that it can look at the whole history $f : (\mathcal{S})^* \to E^\perp$ where $(\mathcal{S})^*$ is the set of finite state sequences. At any time step t during deployment where the agent uses $\pi^\dagger$, the adversary can inject a trigger $f(s_{0:t})$ ("the sticker") into the current state s_t so that the agent perceives a triggered state $s_t^\dagger = s_t + f(s_{0:t})$ and ends up taking a potentially harmful action $a_t^\dagger \sim \pi^\dagger(s_t^\dagger)$ at state s_t. We emphasize that the trigger only applies to the agent's perceived state $s_t^\dagger$; the underlying MDP state is still s_t. Transition and reward happen according to $(s_t, a_t^\dagger)$, not $(s_t^\dagger, a_t^\dagger)$. By carefully design f, the adversary can drastically decrease the value of a policy over a period of time.

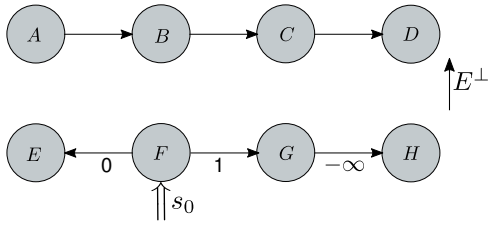


Figure 2: Sanitizing triggered states using subspace projection.

For example, consider the toy MDP given in Figure 2 where states are in $\mathbb{R}^2$. The initial state is always state F. Actions and rewards are given by black arrows; transition is deterministic. Clearly, the optimal policy π^* is to take "left" at F which yield zero return, and follow the arrow in other states since there is no choice. The safe subspace E is horizontal direction. But the non-reachable states A–D give attacker the opportunity to attack. The attacker's backdoor policy is simply $\pi^\dagger = \pi^*$

in this toy example. [1] But when the agent starts at state $s_0 = F$, the adversary injects a trigger $f(F) \in \mathbb{R}^2$ in upward direction along $E^\perp$ so that the agent perceives its state as $s_0^\dagger = F + f(F) = B$ instead. The agent will then choose action $a_0^\dagger =$"right" as directed by its policy $\pi^\dagger(B)$. However, the underlying MDP has its true state at F, so the agent moves to G. The agent may be happy that it received an immediate reward of 1 compared to 0 reward on left. However, the readers can see the agent is now doomed in the long term without further attacks. This example contrasts sharply with backdoor attacks on supervised learning which focuses on instant gratification.

Assumption 3 *The trigger function f is adaptive but the adversary can only inject a trigger in the $E^\perp$ subspace of the state space. Further, the triggered states are B-bounded in expectation as below,*

$$\forall \pi,\, \forall t \in \mathbb{N} \quad \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} [\|(s_t + f(s_{0:t}))\|_2] \leq B$$

This assumption states that at every time step t, the adversary keep the perceived triggered states B-bounded in expectation. Note that this requirement is in expectation, so the adversary can still sometime inject large triggers as long as it keeps the average triggered states bounded.

Recall the trigger acts at a perceptual level to the agent and does not change the true MDP state. Given an agent using the backdoor policy $\pi^\dagger : \mathcal{S} \to \Delta(\mathcal{A})$, and an adversary with trigger function $f : (S)^* \to E^\perp$, the interaction between them in a triggered environment takes place as follows:

Procedure 1 Backdoor Policy Interaction Protocol

Entities: MDP environment $\mathcal{M}$, RL agent running backdoor policy $\pi^\dagger$, adversary with trigger function f.

1: the environment draws $s_0 \sim \mu$.
2: **for** $t = 0, 1, \cdots$ **do**
3: the adversary injects trigger $f(s_{0:t})$ to state s_t (f can be zero, meaning no trigger).
4: the agent observes $s_t^\dagger = s_t + f(s_{0:t})$ and takes the action $a_t^\dagger \sim \pi^\dagger(s_t^\dagger)$.
5: the environment evolves $s_{t+1} \sim P(\cdot|s_t, a_t^\dagger)$, $r_t \sim R(s_t, a_t^\dagger)$

In a clean environment, the trigger function f is always zero. But given a policy π and a trigger function f, the value of the policy π in the triggered environment is given as,

$$V_{\mathcal{M},f}^{\pi} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t + f(s_{0:t})))\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi \circ f(s_{0:t}))\right] = V_{\mathcal{M}}^{\pi \circ f}$$

Here, we note that the trigger function f affects the value only through action selection. Also, since f is adaptive (i.e. it depends on current state and history), the composition of a Markovian policy π with the trigger function f leads to a non-Markovian policy $\pi \circ f$.

Goal of the defender The defender is provided with a backdoor policy $\pi^\dagger$, and has a sample of interactions between $\pi^\dagger$ and the clean environment $\mathcal{M}$. This is realistic in many applications: for example, even if a user does not trust the driving policy $\pi^\dagger$ she downloads, she can test drive the car for a few days in an enclosed driving facility. The goal of the defender is to sanitize the backdoor policy $\pi^\dagger$ such that the sanitized policy performs near-optimally even in the presence of trigger function f.

4 Sanitization Algorithm and Guarantees

We propose Algorithm 2 to sanitize and render backdoor policy harmless. Our sanitization algorithm works in an unsupervised manner by first recovering an estimate of the safe subspace using the clean samples from $d_{\mathcal{M}}^{\pi^*}$ and projecting every states onto this empirical clean subspace to sanitize the state before feeding into the backdoor policy $\pi^\dagger$ (recall the agent is stuck with $\pi^\dagger$ since she does not have the resources to retrain).

[1] This is to keep the example simple. In general, the backdoor policy can differ from π^* on states outside T.

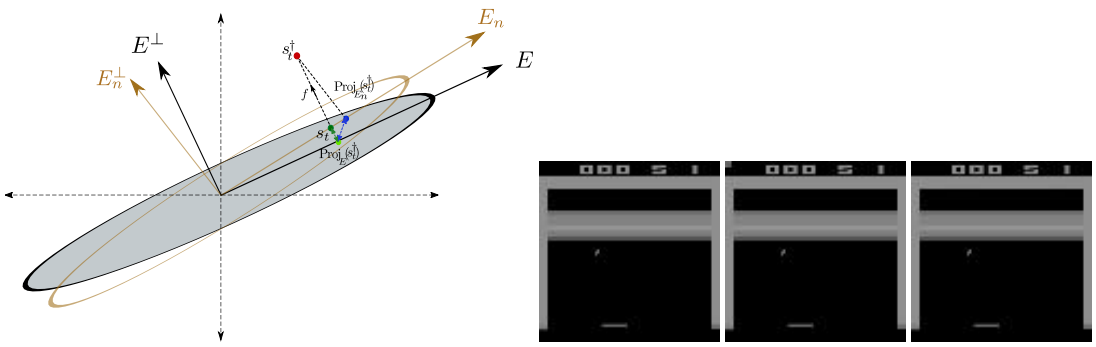
Algorithm 2 Defense through subspace sanitization

Entities: MDP environment $\mathcal{M}$, RL agent with policy $\pi^\dagger$, adversary with trigger function f.
Inputs: sample access to clean environment $\mathcal{M}$, number of clean samples n.

1: **Sanitization phase :**
2: run $\pi^\dagger$ for n clean episodes, collect $\{s_j\}|_{j=1}^n \overset{i.i.d}{\sim} d_{\mathcal{M}}^{\pi^\dagger}$.
3: calculate the empirical covariance $\Sigma_n = \frac{1}{n}\sum_{j=1}^n s_j s_j^\top$ and its eigen decomposition,

$$\Sigma_n = \sum_{i=1}^D \hat{\lambda}_i \hat{u}_i \hat{u}_i^\top \quad \text{where } \hat{\lambda}_1 \geq \cdots \hat{\lambda}_d \geq \hat{\lambda}_{d+1} \geq \cdots \geq \hat{\lambda}_D.$$

4: construct the empirical projection operator $\mathit{Proj}_{E_n} = \sum_{i=1}^d \hat{u}_i \hat{u}_i^\top$.
5: define the sanitized policy $\pi_{E_n}^\dagger : \mathcal{S} \to \Delta(A)$ s.t. $\forall s \in \mathcal{S}$, $\pi_{E_n}^\dagger(s) = \pi^\dagger \circ \mathit{Proj}_{E_n}(s)$.
6: **Deployment phase :**
7: at every time step t, the agents takes the action $\pi_{E_n}^\dagger(s_t^\dagger)$.



(a) Sanitization. (b) Clean state. (c) Triggered state. (d) Sanitized state.

Figure 3: Sanitization. (b,c,d) shows an example from the breakout game. Note that the square trigger (top left) in the triggered state is filtered out after projection onto the empirical safe subspace.

Algorithm 2 has a strong guarantee. We recall that the backdoor policy performs optimally in the clean environment i.e. $V_{\mathcal{M}}^{\pi^\dagger} = V_{\mathcal{M}}^*$. So, we would like the performance of sanitized policy $\pi_{E_n}^\dagger$ in the triggered environment$(\mathcal{M}, f)$ to be as close as to the performance of the optimal policy in the clean environment. Thus, we are interested in bounding the difference between $V_{\mathcal{M}}^*$ - $V_{\mathcal{M}}^{\pi_{E_n}^\dagger \circ f}$ which is stated in the next theorem.

Theorem 4 *Let $\pi^\dagger$ be the backdoor policy and let $d_{\mathcal{M}}^{\pi^\dagger}$ denote the discounted state occupancy distribution of $\pi^\dagger$ under clean environment MDP $\mathcal{M}$. Further, let $\delta_* = \lambda_d - \lambda_{d+1} > 0$ be the eigen gap between the safe subspace E and $E^\perp$. Under assumptions 1, 2, 3 stated above, $\forall \epsilon > 0, \forall \delta > 0$, if $n \geq \frac{CdB^2 L_E^2 K^4 \|\Sigma\|^2}{{\delta_*}^2 (1-\gamma)^4 \cdot \epsilon^2}\left(D + \log\frac{2}{\delta}\right)$, we have with probability $\geq 1-\delta$,*

$$V_{\mathcal{M}}^* - V_{\mathcal{M}}^{\pi_{E_n}^\dagger \circ f} \leq \underbrace{\frac{L_E}{(1-\gamma)^2} \cdot \sqrt{\sum_{i=d+1}^D \lambda_i}}_{\textit{approximation error}} + \underbrace{\epsilon}_{\textit{estimation error}} \tag{3}$$

We make the following remarks:

- The first term in the RHS is a non-negative approximation error ϵ_{app} which cannot be avoided with our algorithm even in the presence of infinite clean samples. Note that this error is zero if the eigen energy of $E^\perp$ subspace is zero, i.e. the adversary chooses to inject triggers along null eigen spaces of Σ.
- The second term is an estimation error and it can be made as small as possible given the number of samples n is chosen large enough.

- If the adversary chooses to hide triggers along high variance dimensions in Σ, it may have to compromise on the performance of the backdoor policy in the clean environment - since the high variance dimensions are often important for underlying task.
- The overall sample complexity is polynomial in effective horizon $\frac{1}{1-\gamma}$, $\frac{1}{\delta_*}$, $\frac{1}{\epsilon}$ and state space dimension D.

Proof The key step in the proof is the following regret decomposition:

$$V^*_{\mathcal{M}} - V^{\pi^\dagger_{E_n}\circ f}_{\mathcal{M}} = V^{\pi^\dagger}_{\mathcal{M}} - V^{\pi^\dagger_{E_n}\circ f}_{\mathcal{M}} \tag{4}$$

$$= \underbrace{V^{\pi^\dagger}_{\mathcal{M}} - V^{\pi^\dagger_E}_{\mathcal{M}}}_{(1)} + \underbrace{V^{\pi^\dagger_E}_{\mathcal{M}} - V^{\pi^\dagger_E\circ f}_{\mathcal{M}}}_{(2)} + \underbrace{V^{\pi^\dagger_E\circ f}_{\mathcal{M}} - V^{\pi^\dagger_{E_n}\circ f}_{\mathcal{M}}}_{(3)} \tag{5}$$

Term (1) leads to the approximation error in the clean environment. Term (2) is the performance difference of the true projected policy $\pi^\dagger_E = \pi^\dagger \circ \mathit{Proj}_E$ when acting on clean vs triggered environment. Term (3) is the estimation error that arises due to sample access to the clean environment.

Since we assumed that the triggers always lie in $E^\perp$ and projection onto E engulfs all the triggers, the second error term reduces to zero. To see this, observe that,

$$\begin{aligned} V^{\pi^\dagger_E\circ f}_{\mathcal{M}} &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, \pi^\dagger(\mathit{Proj}_E(s_t + f(s_{0:t})))) \right] \\ &\overset{(1)}{=} \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, \pi^\dagger(\mathit{Proj}_E(s_t))) \right] = V^{\pi^\dagger_E}_{\mathcal{M}} \end{aligned} \tag{6}$$

where (1) is due to the fact that projection operator in linear and the trigger $f(s_{0:t})$ is always perpendicular to the subspace E, so $\mathit{Proj}_E(f(s_{0:t})) = 0$.

We now bound the approximation error in the clean environment, i.e., term (1) in (5):

$$\begin{aligned} V^{\pi^\dagger}_{\mathcal{M}} - V^{\pi^\dagger_E}_{\mathcal{M}} &\overset{(1)}{=} \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\left[\left(\pi^\dagger(s) - \pi^\dagger_E(s)\right)^\top Q^{\pi^\dagger_E}(s,\cdot)\right] \\ &\overset{(2)}{\leq} \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\left[\|\pi^\dagger(s) - \pi^\dagger_E(s)\|_1 \|Q^{\pi^\dagger_E}(s,\cdot)\|_\infty\right] \\ &\overset{(3)}{\leq} \frac{L_E}{1-\gamma}\cdot \|Q^{\pi^\dagger_E}\|_\infty \cdot \sqrt{\mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\|(I - \mathit{Proj}_E)s\|_2^2} \\ &= \frac{L_E}{1-\gamma}\cdot \|Q^{\pi^\dagger_E}\|_\infty \cdot \sqrt{\mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\|\textstyle\sum_{i=d+1}^{D} u_i u_i^\top s\|_2^2} \\ &\overset{(4)}{=} \frac{L_E}{(1-\gamma)^2}\cdot \sqrt{\textstyle\sum_{i=d+1}^{D}\lambda_i} \end{aligned} \tag{7}$$

where (1) uses the performance difference lemma 8, (2) is Holders inequality, (3) uses L_E-Lipschitzness of $\pi^\dagger$ and Jensen's inequality and (4) follows from the following :

$$\begin{aligned} \mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\|\sum_i u_i u_i^\top s\|_2^2 &= \sum_i \mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\left[u_i^\top s s^\top u_i\right] = \sum_i \mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\left[\mathrm{tr}(s s^\top u_i u_i^\top)\right] \\ &= \sum_i \mathrm{tr}(\mathbb{E}_{s\sim d^{\pi^\dagger}_{\mathcal{M}}}\left[s s^\top\right] u_i u_i^\top) = \sum_i \mathrm{tr}(\lambda_i u_i u_i^\top) = \sum_i \lambda_i \end{aligned} \tag{8}$$

Next, we bound the estimation error in the triggered environment, i.e. term (3) in (5). Let $\pi = \pi^\dagger \circ \mathit{Proj}_E, \pi' = \pi^\dagger \circ \mathit{Proj}_{E_n}$, then the difference in value of these policies in a triggered environment

is given as

$$
\begin{aligned}
& V_{\mathcal{M}}^{\pi \circ f}(s_0) - V_{\mathcal{M}}^{\pi' \circ f}(s_0) \qquad (9) \\
& \overset{(1)}{=} \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} \left[Q_{\mathcal{M}}^{\pi' \circ f}(s_{0:t}, \pi \circ f(s_{0:t})) - Q_{\mathcal{M}}^{\pi' \circ f}(s_{0:t}, \pi' \circ f(s_{0:t})) \right] \\
& = \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} \left[(\pi \circ f(s_{0:t}) - \pi' \circ f(s_{0:t}))^\top Q_{\mathcal{M}}^{\pi' \circ f}(s_{0:t}, \cdot) \right] \\
& \overset{(2)}{\leq} \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} \left[\|\pi \circ f(s_{0:t}) - \pi' \circ f(s_{0:t})\|_1 \|Q_{\mathcal{M}}^{\pi' \circ f}(s_{0:t}, \cdot)\|_\infty \right]
\end{aligned}
$$

Using $\pi = \pi^\dagger(\mathit{Proj}_E(\cdot)), \pi' = \pi^\dagger(\mathit{Proj}_{E_n}(\cdot))$ and using L_E-lipschitzness of $\pi^\dagger$, we obtain

$$
\begin{aligned}
& = \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} \left[\|(\pi^\dagger(\mathit{Proj}_E(s_t + f(s_{0:t}))) - \pi^\dagger(\mathit{Proj}_{E_n}(s_t + f(s_{0:t})))\|_2 \right] \|Q_{\mathcal{M}}^{\pi' \circ f}(\cdot, \cdot)\|_\infty \\
& \overset{(3)}{\leq} \sum_{t=0}^{\infty} \gamma^t L_E \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} \left[\|(\mathit{Proj}_E - \mathit{Proj}_{E_n})(s_t + f(s_{0:t}))\|_2 \right] \|Q_{\mathcal{M}}^{\pi' \circ f}(\cdot, \cdot)\|_\infty \\
& \overset{(4)}{\leq} L_E \cdot \|Q_{\mathcal{M}}^{\pi' \circ f}(\cdot, \cdot)\|_\infty \|\mathit{Proj}_E - \mathit{Proj}_{E_n}\|_2 \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s_{0:t} \sim d_{\mathcal{M}}^{\pi \circ f, 0:t}} [\|(s_t + f(s_{0:t}))\|_2] \\
& \overset{(5)}{\leq} \frac{B \cdot L_E}{(1-\gamma)^2} \|\mathit{Proj}_E - \mathit{Proj}_{E_n}\|_2 \qquad (10)
\end{aligned}
$$

where (1) follows from performance difference lemma 8, (2) follows from Holder's inequality, (3) follows from L_E-Lipschitzness of $\pi^\dagger$, (4) follows from L_2 matrix vector norm inequality and (5) uses the B-boundedness of trigger functions.

Now, if $n \geq \frac{CdB^2 L_E^2 K^4 \|\Sigma\|^2}{{\delta_*}^2 (1-\gamma)^4 \cdot \epsilon^2} \left(D + \log \frac{2}{\delta}\right)$, using Lemma 6 with (10), we can upper bound the estimation error by ϵ:

$$V_{\mathcal{M}}^{\pi_E^\dagger \circ f}(s_0) - V_{\mathcal{M}}^{\pi_{E_n}^\dagger \circ f}(s_0) \leq \epsilon.$$

Finishing the proof of Theorem 4 : We plug (6), (7), (4) together back in (5) to get

$$V_{\mathcal{M}}^{\pi^*} - V_{\mathcal{M}}^{\pi_{E_n}^\dagger \circ f} \leq \frac{L_E}{(1-\gamma)^2} \cdot \sqrt{\sum_{i=d+1}^{D} \lambda_i} \quad + \epsilon. \qquad (11)$$

■

5 Experiments

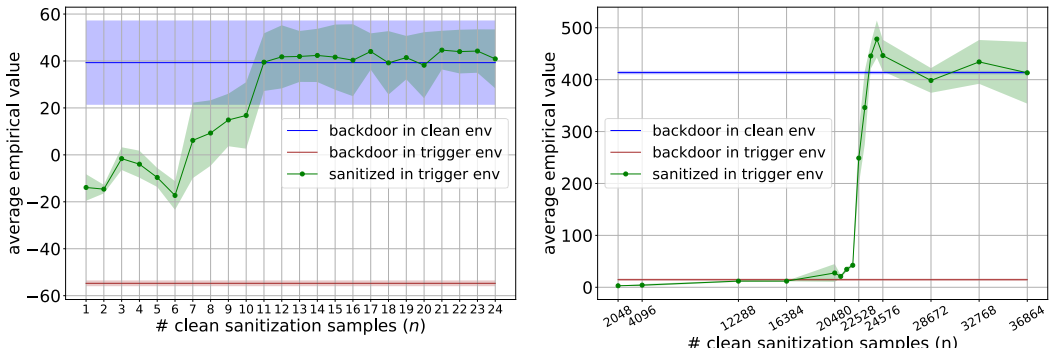
In this section, we present some experimental results that validate our sanitization algorithm against backdoor attacks in Atari game environments. We performed the sanitization experiments on backdoor policies in two games: (i) boxing-ram game which has a vector RAM state space, (ii) breakout game which has a image state space. In each case, we acted as the attacker and pre-trained a backdoor policy. The agent using the backdoor policy does very well in clean environment and poorly in triggered environment. In each case, we used our sanitization algorithm to sanitize the backdoor policy and reported their results.

5.1 Boxing-Ram game

Environment and attack specification The state consists of a 32 dimensional byte RAM vector: a concatenation of four 8-dimensional (4×8) byte RAM vector from last four time steps. The

attacker exploits one of the unused coordinates (which is an always zero in clean environment) to inject triggers($f = 255 \cdot e_{28}$ where e_{28} is 28th canonical basis vector in $\mathbb{R}^{32}$) directly into the vector RAM state space. The action space consists of 18 different types of action - move left, right, up, down, right and kick, down and kick and so on.

Backdoor policy The backdoor policy consists of a 4 layered fully connected neural network with second layer shared to the critic network. The joint actor critic network is trained using PPO algorithm [14] implemented in Pytorch. The attack training strategy utilized a reward manipulation scheme that in triggered environment forces agent to take action leading to low reward. When the agent uses this backdoor policy during deployment and the attacker triggers it, the agent keeps moving close to the opponent to get beaten down without fighting back. While in the clean environment, the backdoor policy consistently performs very well.



(a) Boxing-ram game. (b) Breakout game.

Figure 4: Performance of the backdoor policy in the clean environment(blue), and in the triggered environment(brown). With sufficient clean sanitization samples(n), our sanitized policy recovers back the clean performance in both the instances(green). The solid line represent the mean empirical value of the policy and the shaded region denote standard deviation of it.

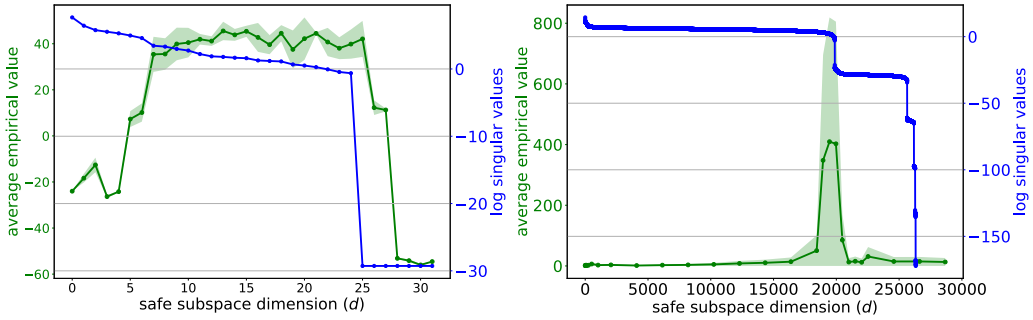
5.2 Breakout game

Environment and the attack specification We consider a version of breakout game where the states consist of a stack of four down-scaled board images $(4 \times 84 \times 84)$ from the last 4 time steps and the action space consists of three actions: move left, move right, and do not move. The attack directly takes place in the image state space where the attacker injects a 6×6 pixel square trigger on the top left part of the arena.

Backdoor policy The backdoor policy is a neural network with two convolutional layer followed by two dense layer with ReLU activation units. The policy has been trained using a Tensorflow implementation of the PAAC algorithm [2]. The policy has been trained so that in absence of triggers i.e. in the clean environment, it performs very well and consistently receives high empirical returns. However, in presence of trigger the policy falters into choosing 'do not move' action, and the agent consistently receives very low empirical returns.

5.3 Empirical results of our sanitization algorithm

For a fixed sanitization sample count n, the defender collects n clean episodes from the environment and chooses an independent sample from each episode to get n samples $d_{\mathcal{M}}^{\pi^\dagger}$. Empirically, we observe that choosing roughly $2n$ correlated samples for each n also works just well for our sanitization purpose. It then constructs a sanitized policy (as defined in algorithm 2) using top d eigen bases of the empirical covariance matrix. The safe space dimension d is decided using the eigen gap in empirical spectrum of covariance matrix(all eigen vectors with eigen values $\geq 10^{-10}$, see section 5.4 for further discussion) are chosen in safe subspace.



(a) Boxing-ram game.

(b) Breakout game.

Figure 5: Performance of sanitized policies with different safe space dimension candidates(in green); Plot of log singular value spectrum of empirical covariance matrix with $n = 40$ (in blue).

In Figure 4 the blue and the brown line shows the performance of backdoor policy in clean and triggered environment respectively. The y-axis shows the mean and standard deviation of empirical value obtained from 4 independent trials. In each trial, we estimated the empirical value from averaging the returns obtained from 5 independent episodes sampled using the respective policies. We observe that the backdoor policy performs pretty well in the clean environment obtaining high value(blue line). However, its performance drops significantly in the presence of triggers(brown line). Next, we show the performance of sanitized policy constructed using n clean samples(on x-axis) from $d^{\pi^\dagger}_{\mathcal{M}}$. We clearly observe that in both the examples, the performance of sanitized policy increases with n and after a sufficient number of clean samples the sanitized policy recovers backs the original clean performance(of backdoor policy in the clean environment - blue line) even when acting in the triggered environment. Thus, our sanitization algorithm works well against the backdoor attacks and requires only a few samples from the clean environment to restore the good performance.

5.4 Dependence of our algorithm on the dimension d of safe subspace

Our theory assumes that the defender knows the dimension of the safe space d, so it can construct a projection operator onto the top-d subspace of the empirical covariance matrix. In practice, the defender can analyze the eigen spectrum of the empirical covariance matrix Σ_n to estimate such a d. A good estimate of d is important, because both underestimation and overestimation can lead to loss in performance. Specifically, with underestimation the defender might lose important dimensions in the safe subspace, and with overestimation the defender may include spurious dimensions from $E^\perp$. This phenomenon is depicted by the empirical value curve (green) in Figure 5, where we see that the performance of the sanitized policy decreases with underestimation or overestimation of d. In practice one can choose d based on the spectral gap of the empirical covariance matrix Σ_n. From the singular value spectrum curve (blue) in Figure 5, we observe that the correct threshold corresponds to dimension just before the first major dip in singular values - which occurs after 24th dimension for boxing-ram game and near $20k$ for breakout game.

6 Conclusion

We formally specified backdoor policy attacks in reinforcement learning. We then proposed a sanitization algorithm that allows a user to safely use a backdoor policy. Our defense has the advantage of being a wrapper method and does not require expensive policy retraining. Our sanitization work makes RL safer and contributes to trustworthy AI. Future work will address the limitations of our defense, namely the need for clean environment interactions and the assumption that triggers reside in a subspace $E^\perp$.

References

[1] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering, 2018.

[2] Alfredo V. Clemente, Humberto Nicolás Castejón Martínez, and Arjun Chandra. Efficient parallel methods for deep reinforcement learning. *CoRR*, abs/1705.04862, 2017.

[3] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. 2019.

[4] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2017.

[5] Junfeng Guo, Ang Li, and Cong Liu. Backdoor detection in reinforcement learning, 2022.

[6] Kiran Karra, C. Ashcraft, and Neil Fendley. The trojai software framework: An opensource tool for embedding trojans into deep learning models. *ArXiv*, abs/2003.07233, 2020.

[7] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: Trojan attacks on deep reinforcement learning agents, 2019.

[8] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks, 2021.

[9] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks, 2018.

[10] Yang Liu, Mingyuan Fan, Cen Chen, Ximeng Liu, Zhuo Ma, Li Wang, and Jianfeng Ma. Backdoor defense with machine unlearning, 2022.

[11] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[12] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans, 2017.

[13] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11957–11965, Apr. 2020.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[15] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks, 2018.

[16] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[17] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.

[18] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Xiaodong Song. Backdoorl: Backdoor attack against competitive reinforcement learning. *ArXiv*, abs/2105.00579, 2021.

[19] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? 2014.

[20] Y. Yu, T. Wang, and R. J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 04 2014.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See abstract and motivation section 1
   (b) Did you describe the limitations of your work? [Yes] See conclusion
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See conclusion
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See assumptions 1,2,3
   (b) Did you include complete proofs of all theoretical results? [Yes] See proof sections in main text and appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] providing an url in supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] providing in supplemental material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the result plots in the main text
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]