PATROL: Provable Defense against Adversarial Policy in Two-player Games

Wenbo Guo[1][*], Xian Wu[2][*], Lun Wang[1], Xinyu Xing[2], Dawn Song[1]
[1] UC Berkeley, [2] Northwestern University
{henrygwb, wanglun, dawnsong}@berkeley.edu, {xianwu2024, xinyu.xing}@northwestern.edu

Abstract

Recent advances in deep reinforcement learning (DRL) takes artificial intelligence to the next level, from making individual decisions to accomplishing sophisticated tasks via sequential decision makings, such as defeating world-class human players in various games and making real-time trading decisions in stock markets. Following these achievements, we have recently witnessed a new attack specifically designed against DRL. Recent research shows by learning and controlling an adversarial agent/policy, an attacker could quickly discover a victim agent's weaknesses and thus force it to fail its task.

Due to differences in the threat model, most existing defenses proposed for deep neural networks (DNN) cannot be migrated to train robust policies against adversarial policy attacks. In this work, we draw insights from classical game theory and propose the first provable defense against such attacks in two-player competitive games. Technically, we first model the robust policy training problem as finding the nash equilibrium (NE) point in the entire policy space. Then, we design a novel policy training method to search for the NE point in complicated DRL tasks. Finally, we theoretically prove that our proposed method could guarantee the lower-bound performance of the trained agents against arbitrary adversarial policy attacks. Through extensive evaluations, we demonstrate that our method significantly outperforms existing policy training methods in adversarial robustness and performance in non-adversarial settings.

1 Introduction

Deep reinforcement learning is one of the heated topics in both academia and industry. Its goal is to learn a policy that controls an agent to take a sequence of actions and fulfill a task in an environment. Guided by the policy, the agent interacts with the environment and receives rewards, indicating how well the agent performs. The policy learning objective is to maximize the cumulative reward of the agent. Based on this objective, existing research proposes the Proximal Policy Optimization (PPO) algorithm [64], which has demonstrated great success in training powerful agents for single-player reinforcement learning tasks (*e.g.,* Atari games [2]).

Going beyond the single-player setup, a more challenging task is the two-player competitive setup, where the goal of both players is to receive more rewards than their opponents. Recent research [5] proposes a self-play mechanism that leverages PPO to train a policy against itself in the two-player game. This framework has dominated the policy training in various RL environments, ranging from simulation games (*e.g.,* MuJoCo [5], Roboschool [55]) to Real-Time Strategy (RTS) games (*e.g.,* Dota 2 [54] and StarCraft II [69]).

Inspired by the great success of RL in two-player games, researchers recently started to exploit its security risk and propose a new adversarial policy attack [25]. Different from other attacks that make an unrealistic assumption (*i.e.,* allowing perturbations to the policy input/output), this attack [25] trains an adversarial policy to interact with the victim player, misleading the victim to take non-optimal actions and forcing the victim to fail its task. Following [25], a recent research [84] proposes another method to train adversarial policies with stronger adversarial exploitability. Due to the differences in problem assumptions, most defenses against attacks in DNNs or other attacks in DRL can not be migrated to defend against adversarial policy/agent attacks [25, 84]. As discussed later in Section 4.1, the only applicable defense (*i.e.,* adversarial retraining) can be easily bypassed by adaptive attacks.

In this work, we take an entirely different design path from existing techniques and propose a novel and provable defense against adversarial policy attacks. Technically, we first transform the robust policy training problem into searching for the optimal policy for both players at the Nash equilibrium point. We analytically show that a policy at the NE point is also a robust policy with a guarantee of its lower-bound performance against arbitrary adversarial policies. Then, leveraging the perturbation-based optimization framework, we design a novel policy training method that optimizes toward the NE point. Finally, we theoretically prove that our method guar-

[*]Equal Contribution.

antees the asymptotic convergence to the NE point and thus guarantees the robustness of the trained policy. Leveraging a distributed learning framework `Ray` [39], we prototype our proposed policy training method and name it as "**Prov**A**ble** defense agains**T** Adversa**R**ial p**OL**icy" (for short `PATROL`).

We extensively evaluate `PATROL` on multiple two-player competitive games, ranging from simple matrix-form and Euclidean games to complicated robot simulators (*e.g.,* MuJoCo [72]) and real-time strategy games (*e.g.,* StarCraft II [69]). We demonstrate that `PATROL` is superior to state-of-the-art policy training methods (i.e., fictitious play [11] and self-play [5]) in the following aspects. First, `PATROL` is more effective in searching for the NE points in different types of games (*e.g.,* games with discrete or continuous action space, games with convex-concave or non-convex non-concave value function). Second, the policies trained by `PATROL` demonstrate stronger capability in non-adversarial settings. Finally, `PATROL` significantly outperforms existing methods in defending against existing adversarial policy attacks. To the best of our knowledge, this is the first work that learns robust policies for different two-player competitive games and the first work that provides a certified robustness guarantee against adversarial policy attacks.

In summary, the paper makes the following contributions.

- We show the policy at the NE point is the robust policy. Guided by this discovery, we propose `PATROL`,[1] a novel robust policy training method.
- We theoretically prove that `PATROL` is guaranteed to converge to the NE point and thus provides certified robust policies against arbitrary adversarial policies, even in games with non-convex non-concave value functions.
- We demonstrate, analytically and empirically, that commonly used defenses are insufficient in providing certified robustness guarantees and thus can be easily bypassed by adaptive attacks.
- We compare `PATROL` with state-of-the-art policy training methods in various environments and demonstrate its superiority in adversarial robustness and performance in non-adversarial settings.

2 Background

As is depicted in Figure 1a, the players in a two-player competitive RL environment observe the current environment state and take action simultaneously. The environment then transits to the next state and rewards each player based on their performance at that step. The goal of both players is to learn an optimal policy that maximizes its long-term reward. Deep reinforcement learning models the player's policy as a DNN, which outputs its next action, given its observation of the current environment. In the following, we will formally model a two-player competitive RL task, followed by the state-of-art method for training DRL policies in such tasks.

[1] https://github.com/Henrygwb/rl_robust_minimax

2.1 Two-player Zero-sum Markov Game

A two-player competitive RL environment is typically modeled as a two-player zero-sum Markov game [91]. Formally, a two-player Markov game is defined as $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}\}^i_{i\in\mathcal{N}}, \mathcal{P}, \{R^i\}_{i\in\mathcal{N}}, \gamma)$, where $\mathcal{N} = 1,2$ denotes the players. $\mathcal{S}$ denotes the state space observed by both players, $\mathcal{A}^i$ represents the action space of player i. Let the joint action $\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2$, $\mathcal{P}: \mathcal{S}\times\mathcal{A} \rightarrow \Delta(\mathcal{S})$ denotes the state transition of the environment. R^i: $\mathcal{S}\times\mathcal{A} \rightarrow \mathbb{R}$ is the reward function for player i. The game is a zero-sum game if $\sum_i R^i(s,a) = 0$ for any state-action pairs, indicating the gain of one player is exactly the loss of the other. $\gamma \in [0,1)$ is the discount factor.

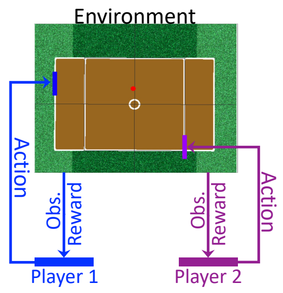
At each time step t, each player takes an action a^i_t based on the current state s_t. Driven by both players' actions, the system then transits to a new state s_{t+1} and rewards each player with an instant reward r^i_t. As mentioned above, the player's goal is to maximize its long-term reward, by learning an optimal policy π^i_*. The long-term reward of the i-th player when the player plays the policy π^i and its opponent plays the policy π^{-i} is defined as the state-value function.

$$V^i_{\pi^i,\pi^{-i}}(s) = \mathbb{E}_{a^i_t\sim\pi^i, a^{-i}_t\sim\pi^{-i}}[\sum_{t\geq 0}\gamma^t R^i(s_t,a_t,s_{t+1})|s_0 = s]\,, \quad (1)$$

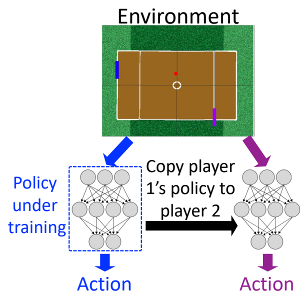
where $a_t = a^i_t \times a^{-i}_t$ represents the joint action. Accordingly, the action-value function of player i is defined as $Q^i_{\pi^i,\pi^{-i}}(s,a) = R^i(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}}[V^i_{\pi^i,\pi^{-i}}(s')]$. Note that a player's state-value and action-value functions depend on not only its own policy but also the other player's policy. This indicates that to obtain optimal performance in a two-player game, a player should choose their policy by considering the choices of their opponent. A common solution for two-player games, *Nash equilibrium*, is defined as [6]

Definition 1 *A Nash equilibrium of a two-player competitive Markov game is a joint policy* $\pi_* = \{\pi^i_*, \pi^{-i}_*\}$*, such that for any* $s\in\mathcal{S}$ *and* $i\in\mathcal{N}$, $V^i_{\pi^i_*,\pi^{-i}_*}(s) \geq V^i_{\pi^i,\pi^{-i}_*}(s)$ *for any* π^i.

NE states that, for each player i, π^i_* is its optimal policy when its opponent plays the policy π^{-i}_*. This also can be expressed as π^i_* is the i-player's *best response* to its opponent's policy π^{-i}_* because $V^i_{\pi^i_*,\pi^{-i}_*}(s) \geq V^i_{\pi^i,\pi^{-i}_*}(s)$. At a NE point, both players are playing their optimal policy and have no incentive to update their policy further. As such, a NE point is a stable and optimal point for both players. For finite games, NE always exists but may not be unique [6]. Note that we typically take the absolute value of the reward when computing the value for the player with a negative reward.



(a) Two-player game.



(b) Self-play mechanism.

Figure 1: Illustrations of with the Roboschool Pong environments [55]. “Obs." stands for observation.

2.2 Self-play with PPO

Researchers in the game theory and RL community have designed various techniques to find NEs in two-player zero-sum Markov games. Most are designed for simple environments with a discrete action space [1, 3, 95] or require accessing the state-transition function [4, 10, 24, 38, 49]. The state-of-art technique for sophisticated environments with continuous action spaces is self-play with policy gradient methods [5, 44, 60].

Self-play. Motivated by the fictitious play method, recent research [66] proposes to train policies in multi-player RL by playing the player against themselves. As demonstrated in Figure 1b, in a two-player environment, the self-play learning process starts by randomly initializing the same policy for both players and selecting a player to train the policy. In each training iteration, it updates the policy of the selected player and then copies the updated (latest) policy to the other player. The training process ends when the winning rates of both players converge to around 50%. Silver *et al.* [66] shows that this training strategy could even defeat professional human players in the GO game. Follow-up research [5, 94] discovers that other than the latest policy, training against randomly selected older versions of the trained policy could further improve the performance. Self-play offers an effective framework for policy training. Under this framework, it is also necessary to decide which algorithm to use for updating the policy in each training iteration. Recent works [5, 12] show that for sophisticated environments, Proximal Policy Optimization (PPO) [21, 64], the state-of-art technique for policy training in single-player RL, provides the highest efficacy.

PPO models a player's policy as a DNN $\pi_\theta(a|s)$, parameterized by θ. To resolve the parameter, PPO proposes the following objective function

$$\begin{aligned}&\text{argmax}_\theta\ \mathbb{E}_{(a_t,s_t)\sim\pi_{old}}[\min(\text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)A_t, \rho_t A_t)]\,,\\ &\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}\,,\quad A_t = A_{\pi_{old}}(a_t, s_t)\,.\end{aligned} \tag{2}$$

Here, π_{old} is the old policy, and $A_\pi(a,s) = Q_\pi(s,a) - V_\pi(s)$ is the advantage function [63]. This objective maximizes the advantage function, which encourages searching for a better policy than π_{old}. To stabilize the training, it also constrains the policy update ratio ρ_t within a certain trust range. By solving Equation (2), a new policy π_θ with less performance variance can be obtained. As mentioned above, integrating PPO into the self-play framework enables decent performance for sophisticated two-player environments (*e.g.,* MuJoCo [5, 21], Roboschool [55], and hide-and-seek [56]).

3 Existing Attacks and Problem Scope

3.1 Existing Attacks in Two-player Games

Threat model. An attacker treats one player as the victim player, fixes its policy, and trains the other player (*i.e.,* adversarial policy/agent) to win the game. This setup simulates a real-world scenario where a game vendor releases default RL agents to play with human professionals or other game bots (*e.g.,* [52, 66]). An attacker trains an adversarial policy to exploit the weakness of the default AI bot and thus decisively win the game. Note that, different from other attacks [29, 62] against DRL, this threat model does not assume attackers have the privilege of manipulating either the environment or the victim agent's policy network. As is discussed in [25, 84], this makes the attack more practical in that an attacker no longer needs to put tremendous effort into hacking the RL engine or varying the physical world. In addition, we follow [25, 84] and set up a black-box attack scenario, where an attacker can only access the public-observable information about a victim (*i.e.,* observation, action, and instant reward) but not its policy network internals. Under this setup, the attacker's goal is to train an adversarial policy to beat the victim in the corresponding game.

PPO-based attack. Gleave *et al.* [25] propose the first attack under this threat model. This attack first fixes the victim policy and treats it as part of the environment for the adversarial player. Then, it uses the aforementioned PPO algorithm to train an adversarial policy. By using this simple method to train adversarial policies, Gleave *et al.* show the obtained policy could defeat the victim player in MuJoCo games [72]. Although an adversarial policy trained by this method sometimes defeats the victim, its overall winning rate is low. As demonstrated in [84], PPO algorithm is originally designed to train a normal policy and has less guidance for one player to identify the weakness of its opponent's policy. As such, directly applying PPO gives a policy with limited exploitability and thus results in a low winning rate.

Action deviation attack. To tackle the limitations above, Wu *et al.* [84] design a novel objective function to train the adversarial policy. This objective function combines the PPO loss (Eqn. (2)) with an action deviation term to explicitly disturb the victim player. This term maximizes the action difference of the victim player with and without facing the adversarial player. The insight is that an adversarial player could influence its opponent's future actions via its own actions. By imposing maximum influence upon the victim's actions, the adversarial player could force the victim to take a series of non-optimal actions and thus reduce its collected reward. As is shown in [84], adversarial policies trained by this attack achieve a much higher attack success rate than those trained by the PPO algorithm [25] in both the MuJoCo You-Shall-Not-Pass game and the Roboschool Pong game [55].

3.2 Our Problem Setup

Assumptions for defenders. First, similar to the attacker, we do not assume the defender (or game developer) could manipulate the environment. As discussed above, this is important for the proposed defense to be physically realistic and generalizable to real-world RL environments. Second, we also do not assume the defender has prior knowledge about the attack it will face. As such, the defender cannot pretrain a defense policy based on a known attack. Third, we do not assume the defender could access or disturb the training process of the adversarial policy. Instead, we allow the defender to update the victim policy periodically after observing a large number of losses. To update the policy, the defender could collect game episodes of the victim player playing against the current or previous opponent and then use the collected episodes to retrain the victim policy network.

With the above assumptions, our goal is to train a defense policy for each player in the two-player game, which guarantees a lower-bound performance against *any unseen* adversarial policies. We also want this policy to preserve its generalizability in non-adversarial settings. This is equivalent to finding a policy with an optimal worst-case performance. Using this policy as the default policy, game developers could guarantee their worst-case reward is bounded when facing arbitrary attacks. Note that we consider the real-world RL environments where the state and action spaces can be discrete or continuous and the state-transition function is unknown.

4 Key Technique

To achieve the goal specified above, we design and develop PATROL . At a high level, we first model robust policy training as finding a Nash equilibrium in a two-player zero-sum game. Then, based on classical game theory, we design a novel policy training algorithm. Finally, we theoretically prove that policies trained by our algorithm are robust against arbitrary adversarial policy attacks. In the following, we start with some naive solutions and discuss their limitations. Then, we introduce our proposed method, followed by the theoretical analysis of the robustness guarantee.

4.1 Possible Solutions and Limitations

Existing research has proposed a large number of adversarial defenses for DNN classifiers. Among these techniques, adversarial retraining [26] and randomized smoothing [18] are the most widely used method for empirical and certifiable defense. Randomized smoothing is not applicable to our problem because our threat model does not allow perturbing the environment. As such, existing defenses [27, 84] under our threat model follow the idea of adversarial retraining. Specifically, suppose the defender (game vendor) releases a policy π^{v} for the player v in a two-player zero-sum game. An attack then launches an adversarial policy π^{α} for the opponent player α, which successfully defeats the victim policy π^{v}. Following adversarial retraining, this defense first plays π^{v} against π^{α} and collects a set of episodes. Then, it retrains π^{v} with the collected episodes. As is shown in [27, 84], the resulted policy $\tilde{\pi}^{v}$ could outperform π^{α}. However, this defense could be easily bypassed by training a new adversarial policy $\tilde{\pi}^{\alpha}$ against $\tilde{\pi}^{v}$, such that $\tilde{\pi}^{\alpha}$ achieves more reward than π^{α} when playing against $\tilde{\pi}^{v}$. Actually, as demonstrated in [27], existing defenses are only effective against the target attack π^{α} but not an arbitrary one (*e.g.,* $\tilde{\pi}^{\alpha}$).

Motivated by this limitation, we propose another possible solution – iteratively adversarial retraining, i.e., iteratively training the victim and adversarial policies against the current adversary and victim. Intuitively, this iterative process may keep searching for better policies for both attacker and victim and thus give a robust policy that could defend against the strongest attacks. After rigorous analysis, we found that this is not a feasible solution. In Appendix A.1, we analytically show that the iteratively adversarial retraining can not guarantee convergence even for simple two-player zero-sum environments. In Appendix B, we further demonstrate that iteratively adversarial retraining is hard to converge, and its policies are still vulnerable to adversarial policy attacks.

4.2 Overview of the Proposed Technique

So far, we have shown that the naive solutions derived from adversarial defenses for DNNs cannot train robust policies. Motivated by these failures, we design a novel defense mechanism. Instead of relying on adversarial retraining, which only finds a local optimal around the currently explored policy space, we draw insight from classical game theory and design a training algorithm to directly find a robust joint policy in the global space. As we will show later in Section 4.4, the joint policy is guaranteed to be robust against arbitrary attacks on both players in two-player zero-sum games. In this section, we provide an overview of our proposed technique.

Insights behind our design. As specified in Section 3, our goal is, for each player, to find a policy with an optimal worst-case (lower-bound) performance against arbitrary adversarial policies trained by existing attacks [25, 84]. This is equivalent to searching for a point in the joint policy space such that one player' policy is the best response (*i.e.,* strongest opponent) to its opponent player's policy. At such a point, for each player, its opponent player's policy is the best response to the player's current policy, meaning the player is playing in the worst-case scenario. At the same time, the player's policy is also the best response to its opponent player's policy, indicating it has already achieved the optimal performance in the worst-case scenario. As such, each player achieves its optimal worst-case performance at this point. Recall that, as stated in Definition 1, at a Nash equilibrium point, the policy of each player is the best response to the other player. As such, *training robust policies with optimal worst-case performance is equivalent to finding a Nash equilibrium point in the corresponding two-player zero-sum game*.

Formally, at a NE point (π^1_*, π^2_*), we have $V^1(\pi^1_*, \pi^2_*) \geq V^1(\pi^1, \pi^2_*)$ and $V^2(\pi^1_*, \pi^2_*) \geq V^2(\pi^1_*, \pi^2)$.[2] Given that $V^1 = -V^2$ (since the game is zero-sum), we have

$$V^1(\pi^1, \pi^2_*) \leq V^1(\pi^1_*, \pi^2_*) \leq V^1(\pi^1_*, \pi^2)\,. \tag{3}$$

The right half of this inequality states that π^2_* is the policy that forces π^1_* to receive the lowest long-term reward, indicating π^1_* is playing against its strongest opponent and thus is in its worst-case scenario. The left half of Eqn. (3) then shows that π^1_* is the policy that receives the highest long-term reward against π^2_*, meaning π^1_* achieves the optimal performance in the worst-case scenario. As such, by playing π^1_* for the 1st player, we could guarantee the player's optimal worst-case performance as $V^1(\pi^1_*, \pi^2_*)$. For the 2nd player, we could derive a similar inequality: $V^2(\pi^1_*, \pi^2) \leq V^2(\pi^1_*, \pi^2_*) \leq V^2(\pi^1, \pi^2_*)$, showing that π^2_* is the robust policy for the 2nd player with the optimal worst-case performance of $V^2(\pi^1_*, \pi^2_*)$.

To further explain why policies at the NE point are robust policies, we again take for example the real-world game scenario mentioned in Section 3. Suppose the game vendor releases π^1_* as the default policy for the 1st player. In our threat model, an attacker will then try to train a policy π^2 to defeat π^1_*. According to Eqn. (3), the best policy the attacker can search for is π^2_*. In other words, the attacker cannot find a stronger opponent for π^1_* other than π^2_*, showing that π^1_*'s worst performance is bounded and thus is robust against adversarial attacks. Similarly, π^2_* is the robust policy for the 2nd player. As such, by finding a NE point with a joint policy (π^1_*, π^2_*), we could achieve a robust policy for both players in the game. Based on the analysis above, we can define a pair of policies (π^i, π^{-i}) as robust policies if they satisfy the condition in Eqn. (3), and the corresponding lower bound performance for each player is $V^i(\pi^i, \pi^{-i})$.

Theoretical foundation for training robust policies. Through the analysis above, we transform the problem of training a robust policy into searching for a NE point in a two-player zero-sum game. To learn a NE point, we seek the theoretical foundation from classical game theory and find the following theorem to guide our training algorithm design.

[2]we denote two players in the game as the 1st player and the 2nd player. $V^1(\pi^1, \pi^2) = V^1_{(\pi^1, \pi^2)}(s)$ is the value function of the 1st player under the joint policy (π^1, π^2).

Theorem 1 (Minimax Theorem [20]) *In any finite, two-player, zero-sum game, at any Nash equilibrium, each player receives a payoff that is equal to both its maximin value and its minimax value.*

Here, the maximin value of the i-th player is $\max_{\pi^i} \min_{\pi^{-i}} V^i(\pi^i, \pi^{-i})$, where π^{-i} is the policy of the opponent player. Maximin value is the highest long-term reward the i-th player could receive without knowing the other player's policy. The minimax value is defined as $\min_{\pi^{-i}} \max_{\pi^i} V^i(\pi^i, \pi^{-i})$. It is the lowest long-term reward the opponent player could force the i-th player to receive without knowing the i-th player's policy. Based on Theorem 1, we can obtain the following corollary.

Corollary 1 *Given a joint policy (π^1_*, π^2_*) of a two-player zero-sum game, as long as the following conditions are satisfied*

$$(\pi^1_*, \pi^2_*) = argmax_{\pi^1} argmin_{\pi^2} V^1(\pi^1, \pi^2)\,, \tag{4}$$

and

$$(\pi^1_*, \pi^2_*) = argmin_{\pi^2} argmax_{\pi^1} V^1(\pi^1, \pi^2)\,. \tag{5}$$

(π^1_*, π^2_*) is the joint policy at a NE point.

The proof of this corollary is straightforward. If the conditions are satisfied, playing the joint policy (π^1_*, π^2_*) could achieve the minimax value and the maximin value of the 1st player. Given that $V^2 = -V^1$, we have $\max_{\pi^1} \min_{\pi^2} V^1(\pi^1, \pi^2)$ equals to $\min_{\pi^1} \max_{\pi^2} V^2(\pi^1, \pi^2)$, showing the maximin value of the 1st player is the minimax value of the 2nd player. Similarly, the minimax value of the 1st player is the maximin value of the 2nd player. As such, (π^1_*, π^2_*) achieves the minimax value and the maximin value for both players. According to Theorem 1, the payoff of each player at any Nash equilibrium point is equal to its minimax value and the maximin value. (π^1_*, π^2_*) corresponds to a joint policy at a NE point.

To better explain the insights behind Corollary 1, we derive Eqn. (3) from the conditions in Corollary 1. Specifically, suppose solving the inner optimization $\min_{\pi^2} V^1(\pi^1, \pi^2)$ of the Eqn. (4) gives a class of policy $\pi^2 = g(\pi^1)$, which is represented as a function of π^1. Since $g(\pi^1) = \text{argmin}_{\pi^2} V^1(\pi^1, \pi^2)$, we have $V^1(\pi^1_\prime, g(\pi^1_\prime)) \leq V^1(\pi^1_\prime, \pi^2)$ for any $\pi^1_\prime$. That is, $g(\pi^1)$ is the class of policies for the 2nd player that forces the 1st player to receive the lowest long-term reward. Then, suppose further solving the outer optimization $\max_{\pi^1} V^1(\pi^1, \pi^2)$ gives π^1_*, Plugging π^1_* into the inequality above, we have $V^1(\pi^1_*, g(\pi^1_*)) \leq V^1(\pi^1_*, \pi^2)$. Similarly, solving the inner optimization $\max_{\pi^1} V^1(\pi^1, \pi^2)$ of the Eqn. (5) gives $h(\pi^2)$, which satisfies $V^1(h(\pi^2_\prime), \pi^2_\prime) \geq V^1(\pi^1, \pi^2_\prime)$ for any $\pi^2_\prime$. Then, suppose $\pi^2_* = \text{argmin}_{\pi^2} V^1(h(\pi^2), \pi^2)$ is the solution of the outer optimization of the Eqn. (5), we have $V^1(h(\pi^2_*), \pi^2_*) \geq V^1(\pi^1, \pi^2_*)$. So far, we have the following inequalities

$$\begin{aligned} V^1(\pi^1_*, g(\pi^1_*)) &\leq V^1(\pi^1_*, \pi^2)\,, \\ V^1(h(\pi^2_*), \pi^2_*) &\geq V^1(\pi^1, \pi^2_*)\,, \end{aligned} \tag{6}$$

where the joint policy $(\pi^1_*, g(\pi^1_*))$ gives the maximin value of the long-term reward and the joint policy $(h(\pi^2_*), \pi^2_*)$ gives the minimax value. If we could achieve that $\pi^1_* = h(\pi^2_*)$ and $\pi^2_* = g(\pi^1_*)$, the joint policy (π^1_*, π^2_*) gives the maximin and minimax value at the same time, i.e., satisfying the conditions in Corollary 1. Besides, when $\pi^1_* = h(\pi^2_*)$ and $\pi^2_* = g(\pi^1_*)$, Eqn. (6) is equivalent to Eqn. (3), indicating (π^1_*, π^2_*) is the policy of a NE point. As such, if a joint policy satisfies the conditions in Eqn. (5) and Eqn. (4), it reaches a NE point and thus is a set of robust policies for both players.

4.3 Proposed Policy Training Algorithm

Corollary 1 provides high-level guidance for learning a NE point in a two-player zero-sum game, i.e., *solving a joint policy that gives the minimax and maximin long-term reward at the same time*. As mentioned in Section 2, existing research has proposed a large body of policy searching and learning methods to find the NE policies in RL environments with a discrete action space and a known state-transition function. However, for sophisticated RL environments with continuous action and state space and an unknown state-transition function, it is still challenging to learn the NE policies. Specifically, there are two challenges: (1) without the state-transition function, we need to find a way to estimate the long-term reward; (2) we need to design a novel optimization framework that resolves the maximin objective function in Eqn. (4) and the minimax objective function in Eqn. (5) at the same time. In this section, we discuss how to tackle these limitations and introduce our proposed policy training algorithm.

Long-term reward estimation. As mentioned in Section 2, the long-term reward of a player is represented as the state-value function defined in Eqn. (1). It is clear that without knowing the state-transition function $\mathcal{P}$, we can compute the exact value of the long-term reward from Eqn. (1). Inspired by the reward approximation methods in existing policy learning methods [73], we propose to approximate the value function with a deep neural network. Specifically, given a joint policy π, we define a DNN $V_\eta(s)$ to approximate the long-term reward of the 1st player in the game. This network takes as input the player's observation of the current state s and outputs the prediction of the player's long-term reward $V^1_\pi(s)$. To train this approximated value function $V_\eta(s)$, we first run the policy in the corresponding environment and collect the ground truth training episodes $\mathcal{T} = \{o^i_{mt}, a^i_{mt}, r^i_{mt}\}_{m=1:M, t=1:T}$, where mt represents the t-th time step in the m-th episodes, T is the total number of time steps in one episode, and M is the total number of episodes. Then, based on the Monte Carlo method [50], we approximate the ground-truth long-term reward for the 1st player at the state s_t as $\tilde{V}_\pi(s_t) = \frac{1}{M}\sum_m \sum_{k=t}^{T} \gamma^k r^1_{mk}$. Using $\tilde{V}_\pi(s_t)$ as the label for $V_\eta(s_t)$, we can learn the parameter of $V_\eta(s)$ by minimizing the following objective function

$$\min_\eta \frac{1}{M}\sum_m \frac{1}{T}\sum_t |V_\eta(s_t) - \tilde{V}_\pi(s_t)|^2 \,. \tag{7}$$

By solving the Eqn. (7) with a gradient-based optimization method, such as `ADAM` [32], we could obtain an approximated value function for the 1st player under the joint policy π. Since the game is zero-sum, the value function of the other player can be simply approximated by $-V_\eta(s)$.

Proposed robust training algorithm. With the approximated value function, we then use it to learn the joint policy at a NE point. Recall that Corollary 1 gives two conditions for achieving a NE point, where each condition is an optimization function. Recent research [13] proves that if a function $f(x,y)$ is convex-concave (convex in x and concave in y) and $-f(x,y)$ is concave-convex (concave in x and convex in y), $\text{argmax}_x\text{argmin}_y - f(x,y) = \text{argmin}_y\text{argmax}_x - f(x,y)$. In our problem, if the value function $V^2(\pi^1,\pi^2)$ (*i.e.,* $V^2(s)$) is convex-concave and $V^1(\pi^1,\pi^2)$ is concave-convex, we can solve only one optimization function (*i.e.,* Eqn. (4) or Eqn. (5)). The resulted policy is the also the solution of the other optimization function and thus is the joint policy at a NE point. However, in sophisticated games with continuous action and state space, the true value function is neither concave nor convex, letting along the function approximated by a DNN. Recent research [31, 94] demonstrates that without the convex-concave property, solving only one optimization function in Corollary 1 cannot guarantee to find a NE point. To reach a NE point from a non-convex non-concave value function, we still need to find a joint policy which satisfies both conditions in Corollary 1.

To achieve this, we borrow the idea from the perturbation-based optimization framework [34, 53]. This framework is original proposed to find a saddle point (x_*, y_*) for an arbitrary function $f(x,y)$, such that $f(x,y_*) \le f(x_*,y_*) \le f(x_*,y)$. In our problem, this is exact the condition at a NE point (Eqn. (3)). At a high level, to find a saddle point, this framework simultaneously solves the minimax and the maximin optimization and reduces the difference between the minimax and maximin value, leading the solutions of both optimizations to converge to the same point. More specifically, it iteratively finds perturbed points u of the current x and v of the current y, such that u is the locally strongest opponent of y and v is the locally strongest opponent of x, and updates x and y against v and u. As we will discuss later, this process minimizes the gap between the minimax and maximin value, approximated by $f(u,y) - f(x,v)$, guiding the optimization process to converge to the saddle point.

Motivated by this framework, we propose a novel policy training algorithm to find a Nash equilibrium in our problem. As specified in Algorithm 1, we first initialize a population of policies for both player (Line 2). In each iteration, for each pair of policies (π^1_{ki}, π^2_{ki}), we first find their strongest opponents from the policy population, i.e., π^2_{vi} and π^1_{ui} (Line 5&6). Since $V^1(\pi^1_{ki}, \pi^2_{vi}) \le V^1(\pi^1_{ki}, \pi^2_{ki})$ and $V^1(\pi^1_{ki}, \pi^2_{ki}) \le V^1(\pi^1_{ui}, \pi^2_{ki})$, we have $V^1(\pi^1_{ki}, \pi^2_{vi}) \le V^1(\pi^1_{ui}, \pi^2_{ki})$. Then, we fix the 2nd player's policy as π^2_{vi} and update π^1_k. After fixing the policy of one player, the Markov game downgrades to a

Algorithm 1: Robust policy training algorithm.

```
1:  Input: Number of iterations $I$, number of inner loop $l$ number of candidate policies $K$, episode size $M$,
2:  Initialize K pairs of policies $(\pi_k^1, \pi_k^2)_{k=1:K}$.
3:  for $i = 1$ to $I$ do
4:      for $k = 1$ to $K$ do
5:          Find the strongest opponent for $\pi_{ki}^1$ from $\{\pi_{\tilde{k}i}^2\}_{\tilde{k}=1:K}$, i.e., $\pi_{vi}^2 = \text{argmin}_{\pi^2} V_{\tilde{k}i}^1(\pi_{ki}^1, \pi_{\tilde{k}i}^2)$
6:          Find the strongest opponent for $\pi_{ki}^2$ from $\{\pi_{\tilde{k}i}^1\}_{\tilde{k}=1:K}$, i.e., $\pi_{ui}^1 = \text{argmax}_{\pi^1} V_{\tilde{k}i}^1(\pi_{\tilde{k}i}^1, \pi_{ki}^2)$
7:          Update $\pi_{ki}^1$ against $\pi_{vi}^2$ using the PPO algorithm with $l$ iterations, i.e., $\pi_{k(i+1)}^1 \leftarrow \text{argmax} V_{ki}^1(\cdot, \pi_{vi}^2)$
8:          Update $\pi_{ki}^2$ against $\pi_{ui}^1$ using the PPO algorithm with $l$ iterations, i.e., $\pi_{k(i+1)}^2 \leftarrow \text{argmin} V_{ki}^1(\pi_{ui}^1, \cdot)$
9:          Update the value function $V_{ki}^1$ by solving Eqn.(7).
10:     end for
11: end for
12: Play each policy in $\{\pi_{kI}^1\}_{k=1:K}$ against each policy in $\{\pi_{kI}^1\}_{k=1:K}$ and select the strongest policy for each party $\pi_{uI}^1$ and $\pi_{vI}^2$.
13: Output: the final policy: $\pi_{uI}^1$ and $\pi_{vI}^2$.
```

single-player MDP, and the policy π_k^1 can be updated by using the state-of-the-art PPO algorithm (Line 7). With the resulted $\pi_{k(i+1)}^1$, we have $V^1(\pi_{ki}^1, \pi_{vi}^2) \leq V^1(\pi_{k(i+1)}^1, \pi_{vi}^2)$. Similarly, we also update π_{ki}^2 against π_{ui}^1 and obtain $\pi_{k(i+1)}^2$, such that $V^1(\pi_{ui}^1, \pi_{k(i+1)}^2) \leq V^1(\pi_{ui}^1, \pi_{ki}^2)$ (Line 8). After running this update process, we could obtain the following inequality from the inequalities above

$$V^1(\pi_{ui}^1, \pi_{k(i+1)}^2) - V^1(\pi_{k(i+1)}^1, \pi_{vi}^2) \leq V^1(\pi_{ui}^1, \pi_{ki}^2) - V^1(\pi_{ki}^1, \pi_{vi}^2). \quad (8)$$

This inequality shows that the gap between $V^1(\pi_u^1, \pi_k^2) - V^1(\pi_k^1, \pi_v^2)$ keeps reducing as the iteration proceeds. Since $V^1(\pi_u^1, \pi_k^2)$ approximates the minimax value in Eqn. (5) and $V^1(\pi_k^1, \pi_v^2)$ approximates the maximin value in Eqn. (4), this optimization process could reduce the difference between these two values and thus push the solution towards a NE point. After solving the new polices, we also update the value function using the estimation method introduced above (Line 9). Finally, we find the strongest policy for each player and output them as the final policies for both players (Line 12).

Note that although both methods leverage the PPO algorithm, our method is fundamentally different from the self-play mechanism in the following aspects. First, rather than training only the policy of one player and copying its policy to the other player, our method updates both players' policies. More importantly, our method updates a current policy against its strongest opponent and pushes the joint solution towards a NE point. As we will discuss in Section 4.4, our method is guaranteed to converge to a NE point, while self-play cannot provide such a robust guarantee. Our empirical evaluation in Section 5 further shows that policies trained by our method are more robust than those obtained by self-play.

It should also be noted that the perturbation-based optimization method mentioned above [34,53] are about general ideas of perturbation-based optimization. We instantiate this idea with our customized designs and enable a novel robust policy learning algorithm, especially for sophisticated two-player RL with unknown state transition, continuous state/action spaces, and non-convex non-concave value functions.

4.4 Theoretical guarantee

In the final part of this section, we provide the theoretical analysis to prove that our proposed training algorithm guarantees to converge to a NE point. In other words, policies trained by our method are provably robust against adversarial policy attacks. Formally, given a pair of polices (π_*^1, π_*^2) obtained by our proposed method, we prove that (π_*^1, π_*^2) asymptotically converges to a NE point, such that $V^1(\pi^1, \pi_*^2) - \epsilon \leq V^1(\pi_*^1, \pi_*^2) \leq V^1(\pi_*^1, \pi^2) + \epsilon$ with $\epsilon \rightarrow 0$.

Recall that the true value function is approximated with a neural network, we first present the following lemma to guarantee that the approximation error is bounded.

Lemma 1 *Given an approximation V_η^1 of V^1 obtained from $M \geq \frac{C}{\epsilon^2} log \frac{2}{p}$ episodes, we have $Pr(|V^1 - V_\eta^1| \leq \epsilon) \geq 1 - p$.*

The proof of this lemma is show in Appendix A.2. It states that with a large number of episodes, the true value function could be accurately approximated by a parametric function with a high probability. With Lemma 1, we then present the convergence guarantee in the following theorem.

Theorem 2 *Under a bounded learning rate, a joint policy (π_*^1, π_*^2) learned from V_η^1 by using Algorithm 1 satisfies the following inequality $V^1(\pi^1, \pi_*^2) - \epsilon \leq V^1(\pi_*^1, \pi_*^2) \leq V^1(\pi_*^1, \pi^2) + \epsilon$, for all π^1 and π^2. V_η^1 is a smooth and differentiable function.*

The proof of this theorem is specified in Appendix A.3. Theorem 2 shows that the solution given by our method guarantees to converge to a ϵ-approximate Nash equilibrium point, where ϵ is the value function approximation error in Lemma 1. As analyzed in Appendix A.2, ϵ keeps decreasing as the number of training episodes increases. Combining this theorem with the analysis in Section 4.2, we theoretically prove that our method learns a joint policy at an approximated Nash equilibrium point and thus provides a robust policy with a guarantee of worst-case performance for each player, even when the value function is non-convex and non-concave.

5 Evaluation

Throughout the evaluation, we seek to answer the following questions: ❶ Can `PATROL` and the existing policy training methods (fictitious play and self-play) find the ground truth NE point in different two-player zero-sum games? ❷ Does PATROL better converge to the unknown NE point than self-play in sophisticated games? ❸ Is PATROL more robust than self-play under the PPO-based attack [25]? ❹ Is PATROL more robust than self-play under the action deviation attack [84]?

5.1 Experiment Setup

Environment selection. To answer question ❶, we select seven toy games: ① a matrix form game with a discrete state/action space, ② two Euclidean games with a continuous state/action space and a convex-concave value function, ③ two Euclidean game with asymmetric action space, ④ two Euclidean game with a continuous state/action space and a non-concave and non-convex value function. All of these games have a unique ground truth Nash equilibrium point. Using these games, we evaluate whether PATROL and selected baseline methods could converge to the NE point in games with different properties: discrete and continuous state/action space (①&②); symmetric and asymmetric action space (②&③); convex-concave and non-convex non-concave value function (②&④). To answer question ❷-❹, we select three types of games – four MuJoCo games [72]: You-Shall-Not-Pass and Kick-And-Defend (with asymmetric action space) and Sumo-Humans and Sumo-Ants (with symmetric action space); One Roboschool Pong game [55] (with symmetric action space); the StarCraft II game [69] (with symmetric action space). All these games are sophisticated RL environments with a continuous state/action space and an unknown non-convex non-concave value function. They are commonly used in academia for evaluating RL algorithms, and self-play with PPO is the state-of-art policy training algorithm in these games. In what follows, we introduce the toy games. Fig. 5 demonstrates the sophisticated games. The action spaces of all selected environments are continuous.

Matrix-form game. We consider the classical matching pennies game [78], where each of the two players has one penny in hand. As the game starts, each player chooses to turn its penny to head or tail. After making their choices, the players reveal their pennies simultaneously. If the pennies match (both heads or tails), the 1st player wins; otherwise, the 2nd player wins. The reward is as follows: $r^1(\mathrm{H},\mathrm{H}) = r^1(\mathrm{T},\mathrm{T}) = 1$, $r^1(\mathrm{H},\mathrm{T}) = r^1(\mathrm{T},\mathrm{H}) = -1$. Here, $r^1(H,T)$ is the reward of the 1st player when it plays head (H), and the 2nd player plays tail (T). $r^2 = -r^1$. This game has a global NE point – $(Pr^1(\mathrm{H}), Pr^1(\mathrm{T})) = (0.5, 0.5)$ and $(Pr^2(\mathrm{H}), Pr^2(\mathrm{T})) = (0.5, 0.5)$.

Euclidean games. The payoff of a Euclidean game is a smooth and differentiable function $f(x,y)$. The 1st player controls x and tries to minimize $f(x,y)$, while the 2nd player aims for maximizing $f(x,y)$ by controlling y. $V^1 = -f(x,y)$ and $V^2 = f(x,y)$. The action space for each player is the domain of the corresponding variable. We consider three types of

Game Type	ID	Value function	Domains	NE point
convex-concave	②-S	$f(x,y)=x^2-y^2-2x$	$x,y\in[-2,2]$	$(1,0)$
symmetric action space	②-C	$f(x,y)=x^2+2xy-4y^2+10x-6$	$x,y\in[-50,50]$	$(-4,-1)$
convex-concave	③-S	$f(x,y)=x^2-2y^2-2xy-6x$	$x\in[-5,5], y\in[-4,4]$	(2, -1)
asymmetric action space	③-C	$f(x,y)=x^2+4xy-2y^2+24x$	$x\in[0,50], y\in[-50,0]$	(-4, -4)
non-convex non-concave	④-S	$f(x,y)=x^2y^2-xy$	$x,y\in[-2,2]$	(0, 0)
symmetric action space	④-C	$f(x,y)=x^3-9x^2-2y^2x^3$	$x,y\in[-50,50]$	(6, 0)

Table 1: The Euclidean games used in our evaluation. "S" and "C" stands for the simple and complicated game.

Euclidean games. For each type, we construct a simple and complicated game following the method in [76]. The simple game has a limited domain and a NE point near (0, 0). The complicated game has a larger domain and a NE point that is relatively far from (0, 0), indicating they are not that easy to be found. Table 1 shows the selected games.

Baseline. As mentioned above, we select the state-of-art policy training method – self-play with PPO as the baseline method. Besides the standard version (denoted as self-play), we also consider two variations: 1) instead of randomly selecting a previous policy as the opponent, we choose the strongest policy as the opponent [74] (denoted as self-play-VA); 2) we replace the PPO with a older but also widely used policy updating method A3C [51] (denoted as self-play-VB). Besides self-play and its variations, we also consider a classical policy learning method – fictitious-play [11]. Because of limited scalability, fictitious-play is not applicable to sophisticated environments. We will compare it with PATROL on toy games.

Evaluation metric. For the toy games, which do not have a winning or losing criterion, we measure the reward of the players. For MuJoCo, Roboschool Pong, and StarCraft II, we measure the winning rate of the players. In addition, we run each policy training process 4 times with different initial states and record the mean results. This process helps remove randomness and access the stability of the training algorithms. When comparing the adversarial robustness, we conduct a paired t-test to compare PATROL with the baseline with the best performance. We report the p-value, where a lower p-value indicates a better statistical significance.

Implementation. Recall that PATROL requires training multiple policies per player, which is computationally more expensive than the self-play mechanism. To make the algorithm implementation more efficient, we leverage a distributed reinforcement learning framework – Ray [39] to implement our proposed algorithm. As is shown in Appendix B.2, with Ray, our training can be completed in a reasonable time. Recall that self-player copies the policy of one player to the other player. It cannot give a policy for each player in one training round. For the games with symmetric action space, we run self-play twice to get a policy for each player. For the games with asymmetric action space, we follow the self-play paper [5] and initialize one policy per player and iteratively update each policy without copying policies. Similar to our method, this modified self-play gives a policy for each player at the same time. As specified in Appendix B.1, we implement our comparison baselines and the two attacks [25, 84] based on their official implementations. For the hyper-parameters shared by PATROL and the baseline methods (*e.g.,* policy model architectures, training iteration, learning rate), we select the widely adopted choices without tailoring for PATROL. We keep them the same for all methods to enable a fair comparison (See Appendix B.1 for more details). The hyper-parameter specific to PATROL is the number of policies per player K. In this evaluation, we fix $K = 2$. Later in Appendix B.2, we perform a sensitivity test on K (which shows PATROL is not that sensitive to the subtle variations in K).

5.2 Experiment Design

Experiment I. To answer question ❶, we use PATROL and four baseline methods (three variations of self-play and fictitious-play) to train a policy for both players in the seven toy games. We compare each method's final policies with the ground truth NE point to verify whether PATROL outperforms these baseline methods in converging to the NE point.

Experiment II. We compare PATROL with three self-play methods in six sophisticated games where the NE point is unknown (Question ❷). First, we train a set of policies in selected sophisticated games using these methods. Next, we play the policy learned by PATROL against the policies learned by the baseline methods for each player in these games, and record the winning rate. Since each player's policy is the strongest response/opponent to the other player's policy at a NE point, the method that converges closer to the NE point should be more difficult to defeat using other policy training methods. As such, we compare the performance of the cross competitions between our policies and those obtained by the baselines to identify the stronger responses and assess the convergence of each method to the NE point. We will also play the policies learned by selected methods against the zoo agents in the MuJoCo and Roboschool Pong games. The policies with higher winning rates demonstrate better convergence to the NE point and greater generalizability.

Experiment III. In this experiment, we compare the adversarial robustness of PATROL and the three self-play methods against the PPO-based attack [25] (Question ❸) in the six sophisticated games. Specifically, for each player in the MuJoCo games, we use the PPO-based attack [25] with the same hyper-parameters to train an adversarial policy against the policy learned by PATROL and baseline methods. Then, we compare the winning rate of the attack policies to investigate which method is more robust against this attack. Finally, for all the policies, we play its policy for one player π^i against its policy of the other player π^{-i} and the corresponding adversarial policy π^{α} and compare the winning rate of π^{-i} with that of π^{α}. If π^{-i} has a higher winning rate, verifying the corresponding method provides a robustness guarantee with a lower-bound performance. As mentioned above, we also conduct a paired t-test to demonstrate the statistical significance of our comparison result.

Experiment IV. To answer question ❹, we design the following experiment. For each player in the six sophisticated games, we use the action deviation attack to train an adversarial policy against the policy learned by PATROL and the self-play methods. Then, we follow the comparisons in experiment III to compare the robustness and verify whether the considered methods could provide a robustness guarantee.

Additional experiments. In addition to the four experiments designed above, we also evaluate the iteratively adversarial retraining defense mentioned in Section 4. We present the experiment results in Appendix B.3, demonstrating this defense cannot even converge to a fixed point on both toy games and sophisticated games. We further demonstrate the robustness of PATROL against a new adversarial policy attack [27] specifically designed for two-player competitive games with non-zero-sum payoffs. Due to the page limit, we also detail this experiment and the corresponding results in Appendix B.4. Finally, as mentioned in Section 5.1, we also conduct an efficiency evaluation and a hyper-parameter sensitivity test and present the results in Appendix B.2.

5.3 Experiment Result

PATROL vs. baseline approaches in toy games. Fig. 2 shows the final policy of the selected methods in seven toy games. As we can see from the figure, none of the baseline approaches could converge to a fixed point in all the settings, where convergence means that the bars of the same method have the same or very similar heights. As mentioned above, the policy in the matching pennies game refers to the probabilities of each player playing head and tail; the policy in the Euclidean games is the value choices of x and y in the value function. For instance, in game ①, none of the baseline methods could converge to a fixed point for the 1st player. Overall, the baseline methods converge in at most 5 out of 14 settings. Even in the cases where self-play converges, it may not converge to the NE point. For example, both self-play and self-play-VA converge to the NE point only in two settings. These results indicate that even in simple non-repeated matrix-form and Euclidean games, the classical fictitious play and self-play-based methods cannot guarantee convergence, let alone converge to the NE point.

In comparison, as is also shown in Fig. 2, PATROL is able to converge to the joint policy at the NE point in all selected games. This result shows that PATROL could consistently search for the NE point when the game properties change and thus verifies the effectiveness of our algorithm design in different setups (discrete/continuous action space, symmetric/asymmetric action space, and convex-concave/non-convex non-concave value function). In addition, it also validates our theoretical analysis in Section 4.4 that PATROL is guaranteed to converge to the NE point even when the value function is non-convex and non-concave. In summary, Fig. 2 demonstrates the superiority of PATROL over the self-play

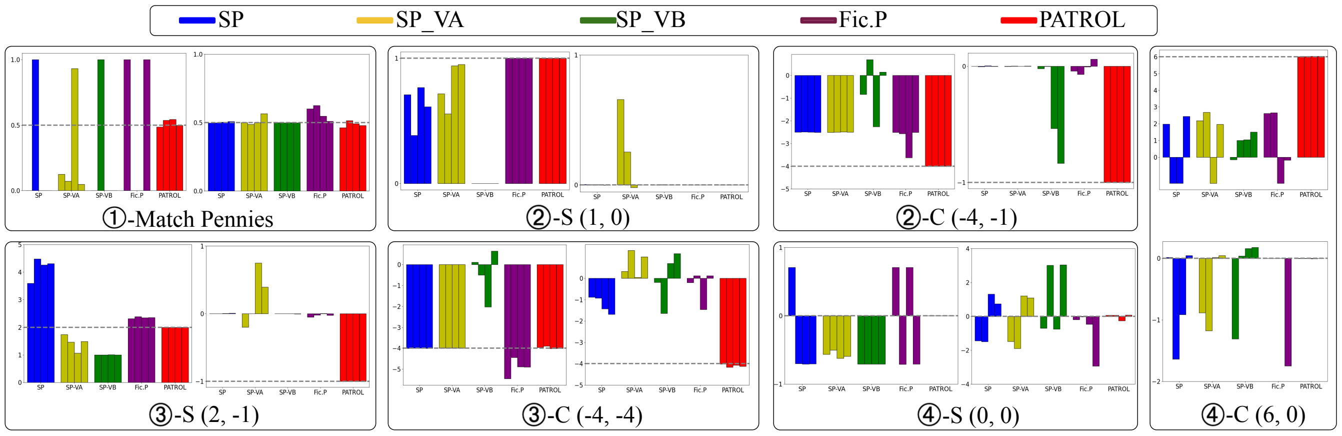


Figure 2: The final policy of PATROL and the comparison baselines in toy games. The left graph in each game is the 1st player. We run each method four times in each setup and draw the converged result of each run as one bar in the figure.

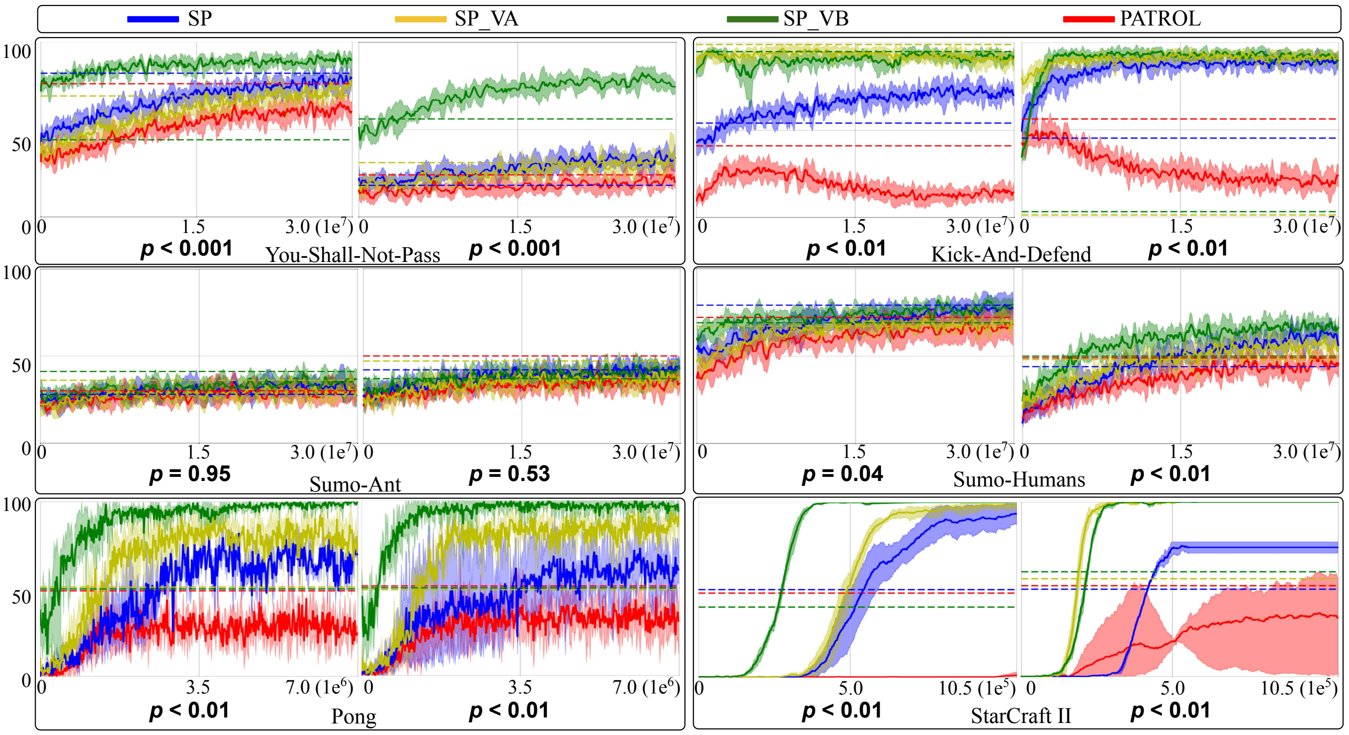


Figure 3: The robustness comparison between PATROL and three baselines against the PPO-based attack. x-axis is the training time step, y-axis is the winning rate. The darker solid lines represent the average winning rates of adversarial policies during the training process, and the lighter bands indicate the variations between the maximal and minimal winning rates. The dotted line is the winning rate of the victim's original opponent trained against the current victim policy. The p-value is the paired t-test result between PATROL and the baseline with the lowest attack winning rate.

mechanism in converging to the NE point. In the following experiments, we will further demonstrate the utilities brought by this superiority in different aspects (*i.e.,* generalizability and adversarial robustness).

PATROL vs. baseline methods in sophisticated games. Table 2 shows the results of our policies against those trained by the original version of self-play. As we can first observe from the upper half table, PATROL_1 has a lower winning rate when playing against PATROL_2 than playing against Self-play_2 (Column 2 vs. 3), indicating PATROL_2 is a stronger response to PATROL_1. Similarly, when playing against PATROL_1, PATROL_2 is a stronger opponent than Self-play_1 (Column 4 vs. 5). This result shows that, for every policy trained by our method, its own opponent policy is a stronger response than the self-play policy. On the contrary, the lower half table shows that when playing against the self-play policies,

Games	PATROL_1 vs.		PATROL_2 vs.	
	PATROL_2	Self-play_2	PATROL_1	Self-play_1
You-Shall-Not-Pass	24.0%	42.0%	76.0%	86.0%
Kick-And-Defend	56.0%	85.0%	41.0%	51.0%
Sumo-Humans	49.0%	55.0%	72.0%	80.0%
Sumo-Ants	50.0%	54.0%	30.0%	40.0%
Pong	51.0%	52.0%	49.0%	53.0%
StarCraft II	52.0%	76.0%	48.0%	74.0%
Games	Self-play_1 vs.		Self-play_2 vs.	
	PATROL_2	Self-play_2	PATROL_1	Self-play_1
You-Shall-Not-Pass	14.0%	18.0%	58.0%	82.0%
Kick-And-Defend	44.0%	45.0%	14.0%	54.0%
Sumo-Humans	22.0%	44.0%	77.0%	79.0%
Sumo-Ants	41.0%	42.0%	28.0%	28.0%
Pong	47.0%	50.0%	48.0%	50.0%
StarCraft II	26.0%	50.0%	24.0%	50.0%

Table 2: The winning rates of policies trained by PATROL vs. self-play policies in six sophisticated games. PATROL_*i* stands for the *i*-player's policy obtained by our method. Self-play_*i* is the *i*-th player's policy trained by self-play. Note that we run each setup four times and report the mean result.

Games	PATROL_1 vs.		PATROL_2 vs.	
	PATROL_2	self-play-VA_2	PATROL_1	self-play-VA_1
You-Shall-Not-Pass	24.0%	40.0%	76.0%	80.0%
Kick-And-Defend	56.0%	89.0%	41.0%	76.0%
Sumo-Humans	49.0%	57.0%	72.0%	73.0%
Sumo-Ants	50.0%	51.0%	30.0%	46.0%
Pong	51.0%	81.0%	49.0%	82.0%
StarCraft II	52.0%	98.0%	48.0%	100.0%
Games	self-play-VA_1 vs.		self-play-VA_2 vs.	
	PATROL_2	self-play-VA_2	PATROL_1	self-play-VA_1
You-Shall-Not-Pass	20.0%	31.0%	60.0%	69.0%
Kick-And-Defend	2.0%	1.0%	11.0%	99.0%
Sumo-Humans	42.0%	48.0%	60.0%	67.0%
Sumo-Ants	41.0%	47.0%	35.0%	36.0%
Pong	18.0%	49.0%	19.0%	51.0%
StarCraft II	0.0%	56.0%	2.0%	40.0%

Table 3: The winning rates of policies trained by PATROL vs. those trained by self-play-VA in six sophisticated games.

PATROL always achieves better performance, indicating the strong responses of self-play policies are policies trained by PATROL rather than their own opponent policies.

Table 3 and 4 further show the performance of our agents playing against those trained by two variations of the self-play, respectively. The results are consistent with those in Table 2. Tables 2-4 reveal that PATROL excels in identifying strong responses for both its own policies and policies trained by three self-play-based baseline methods, indicating that PATROL is superior to these techniques in searching for NE points in sophisticated simulation games.

In Table 5, we show that our policy has a higher winning rate than the baselines when playing against the same zoo agent in the selected games except StarCraft II (because we do not find a zoo agent for StarCraft II). Together with the results in Table 2-4, we can also conclude that policies obtained by PATROL have stronger generalizability than the policies of baseline methods when playing against normal policies.

Defending against the PPO-based attack. Fig. 3 shows the winning rate of adversarial policies trained against PATROL

Games	PATROL_1 vs.		PATROL_2 vs.	
	PATROL_2	Self-play-VB_2	PATROL_1	Self-play-VB_1
You-Shall-Not-Pass	24.0%	79.0%	76.0%	90.0%
Kick-And-Defend	56.0%	94.0%	41.0%	73.0%
Sumo-Humans	49.0%	68.0%	72.0%	86.0%
Sumo-Ants	50.0%	58.0%	30.0%	46.0%
Pong	51.0%	96.0%	49.0%	96.0%
StarCraft II	52.0%	94.0%	48.0%	98.0%
Games	Self-play-VB_1 vs.		Self-play-VB_2 vs.	
	PATROL_2	Self-play-VB_2	PATROL_1	Self-play-VB_1
You-Shall-Not-Pass	10.0%	56.0%	21.0%	44.0%
Kick-And-Defend	6.0%	3.0%	5.0%	95.0%
Sumo-Humans	17.0%	50.0%	55.0%	69.0%
Sumo-Ants	36.0%	37.0%	27.0%	41.0%
Pong	4.0%	50.0%	4.0%	50.0%
StarCraft II	2.0%	60.0%	6.0%	40.0%

Table 4: The winning rates of polices trained by PATROL vs. those trained by Self-play-VB in six sophisticated games.

Games	Zoo_1 vs.				Zoo_2 vs.			
	PATROL_2	Self-play_2	self-play-VA_2	Self-play-VB_2	PATROL_1	Self-play_1	self-play-VA_1	Self-play-VB_1
You-Shall-Not-Pass	13.0%	28.0%	18.0%	62.0%	32.0%	38.0%	46.0%	62.0%
Kick-And-Defend	12.0%	52.0%	84.0%	62.0%	30.0%	50.0%	98.0%	84.0%
Sumo-Humans	14.0%	26.0%	55.0%	68.0%	21.0%	52.0%	65.0%	81.0%
Sumo-Ants	24.0%	25.0%	26.0%	40.0%	21.0%	22.0%	28.0%	28.0%
Pong	48.0%	54.0%	92.0%	98.0%	47.0%	52.0%	88.0%	93.0%

Table 5: The winning rates of a zoo agent against polices trained by the selected methods in five sophisticated games.

and three self-play variations on six sophisticated games. The results show that the adversarial policy trained against our policy has a lower winning rate than those trained against the baseline methods. These results confirm that PATROL is more robust than the baselines against the PPO-based attack. The results in Sumo-Ants are less distinguishable. As discussed in [25], due to limited attack space, adversarial policy attack in this game is naturally hard to succeed. Note that when training an adversarial policy PATROL and baselines in the same setup, we only change the victim policy and keep all the hyper-parameters the same. This ensures that the observed differences in attack performance are because of the choice of victim policy rather than the attack hyperparameters.

Fig. 3 also shows, for victim policies given by PATROL, the mean attack winning rate is consistently lower than the winning rate of the victim's original opponent in all setups. This result shows the PPO-based attack cannot find a stronger response for our policy than its original opponent policy, indicating PATROL provides a lower-bound performance for its trained policy against the PPO-based attack. In contrast, the blue lines in Fig. 3 show that the mean attack winning rate is higher than the winning rate of the original opponent for self-play victim policies, indicating self-play cannot provide a robustness guarantee. Similarly, we also observe that the two variations of self-play fail to provide a robustness guarantee.

Defending against the action deviation attack. Fig. 4 depicts the attack success rate of the action deviation attack against policies trained by PATROL and the baseline methods. Similar to Fig. 3, our method forces lower attack winning rates on all games, verifying PATROL's superiority in robustness against the action deviation attack. Additionally, as shown in Figure 4, despite having a stronger exploitability than the PPO-based attack, the action deviation attack still

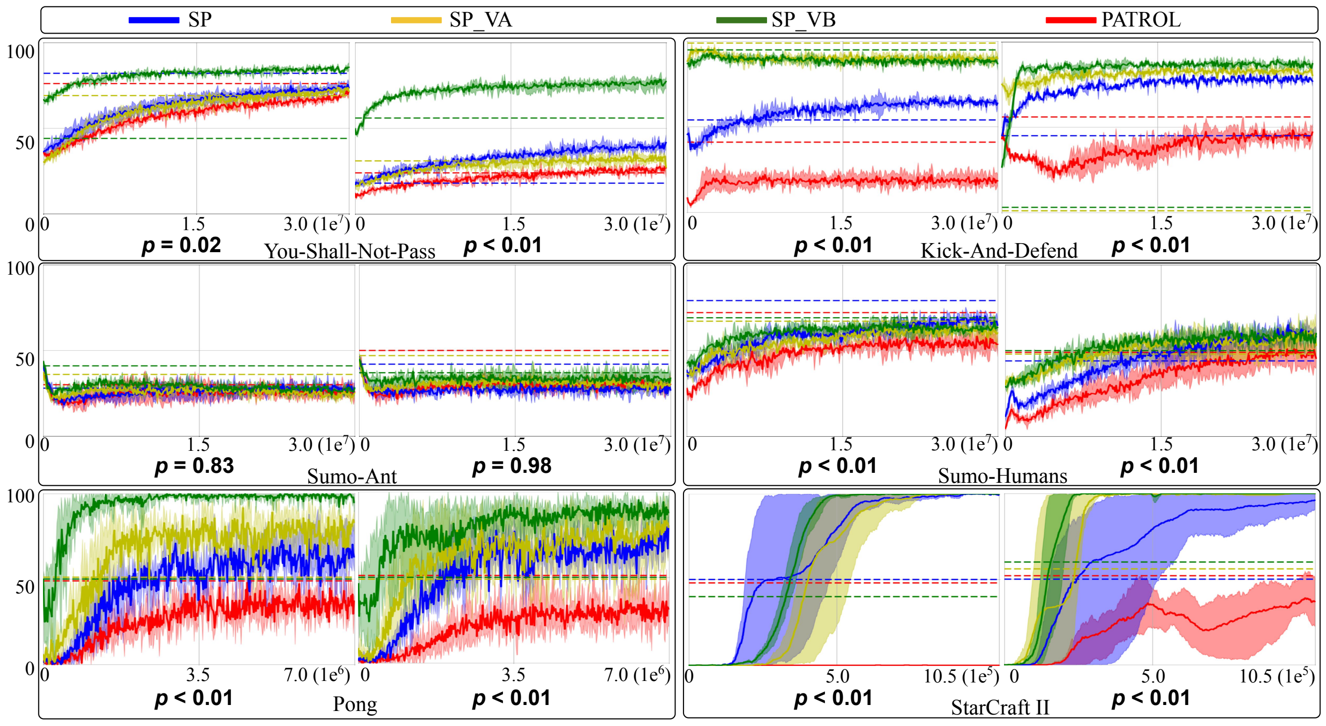


Figure 4: The robustness comparison of PATROL and baselines against the action deviation attack in sophisticated games.

cannot train a stronger response to our policy. This result further confirms our theoretical analysis in Section 4.4 that our method is guaranteed to search for a robust policy at the NE point with a certified lower-bound performance. Together with the result in Fig. 3, we can safely conclude that PATROL is more resilient to existing adversarial policy attacks than the self-play mechanism and its variations.

Robustness vs. generalizability. Our experiments indicate that PATROL exhibits better convergence to a NE point than the baseline methods, which enhances both robustness and generalizability. It is commonly believed that robustness and generalizability are typically a trade-off. However, some counterexamples exist, such as adversarial retraining, which is a widely-used defense mechanism in supervised learning that acts as regularization and can improve the generalizability of the classifier [26]. Similarly, defense distillation [58] has been shown to improve the generalizability of models on certain datasets. We have made a similar observation for PATROL. Nevertheless, as with adversarial retraining, this benefit comes at the cost of computational efficiency.

Qualitative studies. We show the videos of the agent trained by PATROL against its adversarial agents and the agent trained by PATROL against the adversarial agents trained against the self-play in the selected MuJoCo games in https://tinyurl.com/y4c9hdm9. In general, the adversarial agent trained against PATROL's agent no longer establishes weird behaviors, indicating PATROL's agent does not contain obvious defeats that can be exploited by the adversarial attack. In contrast, the adversarial agent against self-play still takes weird actions, but PATROL's agent ignores these actions.

6 Other Related Work

Adversarial perturbation-based attacks. As mentioned in Section 1, prior to adversarial policy attacks, researchers borrow the idea from the adversarial attacks against DNNs (*e.g.*, [16, 19, 43]) and perturb the environment to trigger the failure of a victim agent. Specifically, some pioneer works [7, 29] leverage attacks against DNN [16, 47] to add an adversarial perturbation to the environment at each time step and force the victim player to fail in a single-player environment. Follow-up works explore improving the attack efficiency [35, 41, 62, 68] or practicability [85, 93]. More recent works [42, 70, 88] design techniques to enable optimal attacks under this threat model. Going beyond single-player games, Lin *et al.* [40] further extend the above adversarial environment attacks to multi-player collaborative games. In this work, we remove the attacker's privilege of manipulating the environment, which is more realistic and cost-effective.

Data poisoning attacks. Another line of research generalizes data poisoning attacks [14, 15, 65] to DRL. Specifically, pioneer works [45, 46, 92] poison a player's training episodes by manipulating its reward and thus training non-optimal policies. Yang *et al.* [86] and Kiourti *et al.* [33] further leverage environment and reward manipulation to implant a backdoor into a policy network. During the testing phase, when a trigger is presented in the environment, the victim policy establishes ill-defined behaviors. Most recent work [75] design a trojan attack against two-player games. Rather than perturbing the environment or reward, this attack designs the trigger as an adversarial player's action. The victim player behaves abnormally whenever the adversarial player takes the trigger actions. Since we do not assume access to a victim policy's training process, we do not consider trojan attacks.

Existing defenses. Pioneer defense works [8, 9, 48, 59] adopt adversarial retraining [26] to defend against adversarial environment attacks. By retraining the victim policy in the perturbed environment, this defense improves the policy's resilience against environmental manipulations. More recent works [27, 84] extend this defense to our threat model and propose to robustify a victim policy by retraining it against the corresponding adversarial policy. As discussed in Section 4, this defense can be easily bypassed by training a new adversarial policy against the robustified victim. Besides adversarial retraining, some other recent research [36, 81–83, 89] extends existing certification techniques to provide certified robustness against adversarial environment or data poisoning attacks. Due to different threat models and defense goals, these techniques cannot provide certified defenses in our problem. A final line of works [22, 61, 87] leverage game theory and maximin optimization to defend against adversarial perturbation attacks in single-player RL tasks. They introduce an attack player to control the environment perturbations and transform the original environment into a two-player competitive game between the original RL player and the attack player. They train a robust policy for the RL player by solving a maximin optimization: $\max_{\pi^\nu} \min_{\pi^\alpha} V^\nu(s)$, where ν and α is the RL and the attack player. As discussed in Section 4.2, solving only the maximin or minimax optimization cannot guarantee the convergence to a NE point and thus cannot provide a robustness guarantee in our problem.

7 Discussion

Other defeats of Self-play. In addition to lacking robustness, existing works also discuss other deficiencies of self-play. For example, Balduzzi *et al.* [4] mentioned that self-play cannot establish transitivity. Jaderberg *et al.* [30] pointed out that self-play is limited in emerging human-like behaviors. While our focus differs from these works, it is important to note these additional limitations of self-play.

About Our Robustness Guarantee. As discussed in Section 4, PATROL guarantees its obtained agent has a lower bound against arbitrary adversarial agents (*i.e.,* $V^i(\pi^1_*, \pi^2_*)$ in Eqn. (3)), which varies game by game. We cannot guarantee it is larger than a specific winning rate. This is similar to certain certified defenses for supervised learning [18, 80, 90], which certify their model has a lower-bound accuracy but cannot guarantee it is higher than a specific value.

Extension to extensive-form games. Another two-player competitive game widely existing in the real world is the extensive-form game (*e.g.,* Go [66]), Poker [37]). Different from the Markov game, where two players take action simultaneously, in an extensive-form game, each player takes turns observing the state and taking action. Although no work has done this, as discussed in [27], it is not that hard to extend the existing adversarial policy attacks to this game. Similarly, extending PATROL to extensive-form games and providing an empirical defense is also straightforward. We can just replace the PPO algorithm in Algorithm 1 with a state-of-the-art policy training method in the extensive-form game (*e.g.,* counterfactual regret minimization [23]). However, due to differences in game setup, the robustness guarantee of our method no longer holds for the extensive-form game. Our future work will investigate redesigning PATROL to provide such a guarantee for extensive-form games.

Generalization to multi-player environments. Going beyond two-player games, some DRL tasks involve multiple players, such as Dota [54] and StarCraft [69]. In such games, the players are assigned to two teams. Players of the same team cooperate to compete against the other team. Generalizing PATROL to a multi-player environment is challenging because the dependencies between players are much more complicated than in a two-player game. In the future, we will again draw insights from the game theoretical theorems [57] about finding NE in a multi-player game and extend PATROL to train robust policies in this setup.

Limitation and future work. Our work has a few limitations. First, our method introduces additional computational cost compared to the state-of-art approach. In this work, we improve the efficiency by optimizing our implementation. Our future work will explore further accelerating the training process by selecting better initial states (*e.g.,* using policies pretrained by self-play as the initial policies). Second, we assume the value function is smooth and differentiable. In some special cases, the value functions are non-smooth [67]. Our future work will also investigate adapting our method to these games. Finally, besides the Markov game, some recent works [17, 28] also model a two-player competitive environment as a Stackelberg Game and develop policy training methods with better stability than self-play with PPO. In our future work, we will also explore whether generalizing our method to this model could provide a more stable algorithm.

8 Conclusion

We present PATROL, the first provable defense against adversarial policy attacks in two-player competitive games. Technically, we propose a novel policy training algorithm to find the NE point, which provides a robust policy for each player. We theoretically prove that PATROL is guaranteed to find the NE point in complicated RL environments. Empirical evaluation shows that PATROL outperforms existing policy training methods in finding the NE point, policy generalizability, and defending against existing attacks. With the theoretical analysis and empirical results, we conclude that a policy training method converging to the NE point provides policies with a robustness guarantee against adversarial policy attacks.

Acknowledgments

We would like to thank our anonymous shepherd and reviewers for their helpful feedback. This work was supported in part by the National Science Foundation and the Berkeley Center for Responsible, Decentralized Intelligence.

References

[1] Asma Al-Tamimi, Frank L Lewis, and Murad Abu-Khalaf. Model-free q-learning designs for linear discrete-time zero-sum games with application to h-infinity control. *Automatica*, 2007.

[2] ATARI. Atari games. https://www.atari.com/, 2006.

[3] Yu Bai and Chi Jin. Provable self-play algorithms for competitive reinforcement learning. In *ICML*, 2020.

[4] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *ICML*, 2019.

[5] Trapit Bansal, Jakub Pachocki, Szymon Sidor, et al. Emergent complexity via multi-agent competition. In *ICLR*, 2018.

[6] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[7] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *MLDM*, 2017.

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

[9] Vahid Behzadan and Arslan Munir. Mitigation of policy manipulation attacks on deep q-networks with parameter-space noise. In *SAFECOMP*, 2018.

[10] Michel Benaïm, Josef Hofbauer, and Sylvain Sorin. Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 2005.

[11] Ulrich Berger. Brown's original fictitious play. *Journal of Economic Theory*, 2007.

[12] Shalabh Bhatnagar, Mohammad Ghavamzadeh, Mark Lee, and Richard S Sutton. Incremental natural actor-critic algorithms. In *NeurIPS*, 2008.

[13] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. In *USENIX Security*, 2021.

[15] Nicholas Carlini. Poisoning the unlabeled dataset of semi-supervised learning. In *USENIX Security*, 2021.

[16] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *S&P*, 2017.

[17] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *ICML*, 2022.

[18] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[19] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *Prof. of USENIX Security Symposium*, 2019.

[20] Ding-Zhu Du and Panos M Pardalos. *Minimax and applications*. Springer Science & Business Media, 1995.

[21] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, et al. Implementation matters in deep rl: A case study on ppo and trpo. In *ICLR*, 2020.

[22] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. In *ICLR*, 2022.

[23] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Stochastic regret minimization in extensive-form games. In *ICML*, 2020.

[24] Drew Fudenberg and David K Levine. Consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 1995.

[25] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *ICLR*, 2020.

[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[27] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. Adversarial policy learning in two-player competitive games. In *ICML*, 2021.

[28] Peide Huang, Mengdi Xu, Fei Fang, and Ding Zhao. Robust reinforcement learning as a stackelberg game via adaptively-regularized adversarial training. *arXiv preprint arXiv:2202.09514*, 2022.

[29] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *ICLR workshop*, 2017.

[30] M Jaderberg, WM Czarnecki, I Dunning, L Marris, G Lever, AG Castaneda, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. arxiv. *arXiv preprint arXiv:1807.01281*, 2018.

[31] Parameswaran Kamalaruban, Yu-Ting Huang, Ya-Ping Hsieh, Paul Rolland, Cheng Shi, and Volkan Cevher. Robust reinforcement learning via adversarial training with langevin dynamics. In *NeurIPS*, 2020.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: Trojan attacks on deep reinforcement learning agents. *arXiv preprint arXiv:1903.06638*, 2019.

[34] Galina M Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 1976.

[35] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. In *ICLR Workshop*, 2017.

[36] Aounon Kumar, Alexander Levine, and Soheil Feizi. Policy smoothing for provably robust reinforcement learning. In *ICLR*, 2022.

[37] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[38] Zifan Li and Ambuj Tewari. Sampled fictitious play is hannan consistent. *Games and Economic Behavior*, 2018.

[39] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *ICML*, 2018.

[40] Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *DLS Workshop*, 2020.

[41] 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. In *IJCAI*, 2017.

[42] Guanlin Liu and Lifeng Lai. Provably efficient black-box action poisoning attacks against reinforcement learning. In *NeurIPS*, 2021.

[43] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. {SLAP}: Improving physical adversarial examples with {Short-Lived} adversarial perturbations. In *Prof. of USENIX Security Symposium*, 2021.

[44] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*, 2017.

[45] Thodoris Lykouris, Max Simchowitz, Aleksandrs Slivkins, and Wen Sun. Corruption robust exploration in episodic reinforcement learning. *arXiv preprint arXiv:1911.08689*, 2019.

[46] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. In *NeurIPS*, 2019.

[47] Aleksander Madry, Aleksandar Makelov, et al. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[48] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *IROS*, 2017.

[49] Stephen McAleer, John B Lanier, Roy Fox, and Pierre Baldi. Pipeline psro: A scalable approach for finding approximate nash equilibria in large games. In *NeurIPS*, 2020.

[50] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 1949.

[51] 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 *ICML*, 2016.

[52] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 2017.

[53] Angelia Nedić and Asuman Ozdaglar. Subgradient methods for saddle-point problems. *Journal of optimization theory and applications*, 2009.

[54] OpenAI. Openai at the international 2017. https://openai.com/the-international/, 2017.

[55] OpenAI. Roboschool: open-source software for robot simulation. https://openai.com/blog/roboschool/, 2017.

[56] OpenAI. Emergent tool use from multi-agent interaction. https://openai.com/blog/emergent-tool-use/, 2019.

[57] Guillermo Owen. *Game theory*. Emerald Group Publishing, 2013.

[58] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *S&P*, 2016.

[59] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *AAMAS*, 2018.

[60] Julien Perolat, Bilal Piot, and Olivier Pietquin. Actor-critic fictitious play in simultaneous move multistage games. In *AISTAT*, 2018.

[61] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *ICML*, 2017.

[62] Alessio Russo and Alexandre Proutiere. Optimal attacks on reinforcement learning policies. *arXiv preprint arXiv:1907.13548*, 2019.

[63] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

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

[65] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *USENIX Security*, 2021.

[66] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[67] Leo K Simon. Games with discontinuous payoffs. *The Review of Economic Studies*, 1987.

[68] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *AAAI*, 2020.

[69] Peng Sun, Xinghai Sun, Lei Han, Jiechao Xiong, Qing Wang, Bo Li, Yang Zheng, Ji Liu, Yongsheng Liu, Han Liu, et al. Tstarbots: Defeating the cheating level builtin ai in starcraft ii in the full game. *arXiv preprint arXiv:1809.07193*, 2018.

[70] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. In *ICLR*, 2022.

[71] Richard L Tenney and Caxton C Foster. Non-transitive dominance. *Mathematics Magazine*, 1976.

[72] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *ICIRS*, 2012.

[73] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

[74] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

[75] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoorl: Backdoor attack against competitive reinforcement learning. In *IJCAI*, 2021.

[76] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. *arXiv preprint arXiv:1910.07512*, 2019.

[77] Wikipedia. Hoeffding's inequality. https://en.wikipedia.org/wiki/Azuma%27s_inequality, 2022.

[78] Wikipedia. Match pennies game. https://en.wikipedia.org/wiki/Matching_pennies, 2022.

[79] Wikipedia. Universal approximation theory. https://en.wikipedia.org/wiki/Universal_approximation_theorem, 2022.

[80] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

[81] Fan Wu, Linyi Li, Zijian Huang, Yevgeniy Vorobeychik, Ding Zhao, and Bo Li. Crop: Certifying robust policies for reinforcement learning through functional smoothing. In *ICLR*, 2022.

[82] Fan Wu, Linyi Li, Chejian Xu, Huan Zhang, Bhavya Kailkhura, Krishnaram Kenthapadi, Ding Zhao, and Bo Li. Copa: Certifying robust policies for offline reinforcement learning against poisoning attacks. In *ICML*, 2022.

[83] Junlin Wu and Yevgeniy Vorobeychik. Robust deep reinforcement learning through bootstrapped opportunistic curriculum. In *ICML*, 2022.

[84] Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *Prof. of USENIX Security Symposium*, 2021.

[85] Chaowei Xiao, Xinlei Pan, Warren He, Jian Peng, Mingjie Sun, Jinfeng Yi, Bo Li, and Dawn Song. Characterizing attacks on deep reinforcement learning. *arXiv preprint arXiv:1907.09470*, 2019.

[86] Zhaoyuan Yang, Naresh Iyer, et al. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972*, 2019.

[87] Jing Yu, Clement Gehring, Florian Schäfer, and Animashree Anandkumar. Robust reinforcement learning: A constrained game-theoretic approach. In *ICML*, 2021.

[88] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *ICLR*, 2021.

[89] Huan Zhang, Hongge Chen, Chaowei Xiao, et al. Robust deep reinforcement learning against adversarial perturbations on state observations. In *NeurIPS*, 2020.

[90] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019.

[91] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.

[92] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *ICML*, 2020.

[93] Yiren Zhao, Ilia Shumailov, Han Cui, Xitong Gao, Robert Mullins, and Ross Anderson. Blackbox attacks on reinforcement learning agents using approximated temporal information. *arXiv preprint arXiv:1909.02918*, 2019.

[94] Yuanyi Zhong, Yuan Zhou, and Jian Peng. Efficient competitive self-play policy optimization. *arXiv preprint arXiv:2009.06086*, 2020.

[95] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *NeurIPS*, 2007.

A Additional Technical Analysis

A.1 Iterative Adversarial Retraining

Iterative adversarial retraining expresses the following process. Given a pair of initial policies (π_0^v, π_0^α), iteratively retraining π_t^v against π_t^α and retraining π_t^α against π_{t+1}^v. After T rounds, we can obtain a series of polices for the victim player $\{\pi_0^v, .., \pi_T^v\}$ and the adversarial player $\{\pi_0^\alpha, .., \pi_T^\alpha\}$.

First, we show that this method cannot converge to a fixed point even in a simple matrix form game. Consider the Matching pennies game in Section 5.1, without loss of generalizability, we treat the 1st player as the victim player with a policy π_0^v, where $Pr^v(\mathrm{H}) = p_0^v$ and $Pr_0^v(\mathrm{T}) = 1 - p_0^v$. The attacker trains an adversarial policy π_0^α for the 2nd player where $Pr^\alpha(\mathrm{H}) = p_0^\alpha$. According to [25, 84], the adversarial training solves the following objective function: $\mathrm{argmax}_{p_0^\alpha}[-p_0^v p_0^\alpha + (1-p_0^\alpha)p_0^v + p_0^\alpha(1-p_0^v) - (1-p_0^v)(1-p_0^\alpha)]$ and the optimal solution is $p_0^\alpha = 1$ if $p_0^v \leq 0.5$ and $p_0^\alpha = 0$ if $p_0^v > 0.5$. Suppose $p_0^v \leq 0.5$ and $p_0^\alpha = 1$, retraining the victim policy against π_0^α solves the following optimization: $\mathrm{argmax}_{p_1^v}[p_1^v p_0^\alpha - (1-p_0^\alpha)p_1^v - p_0^\alpha(1-p_1^v) + (1-p_1^v)(1-p_0^\alpha)]$, with the optimal $p_1^v = 1$. Then, we can solve these optimizations to update $p_1^\alpha = 0$, $p_2^v = 0$, $p_2^\alpha = 1$, etc. As we can observe from the above analysis, p_t^v and p_t^α switch between 0 and 1 and thus cannot converge to a fixed point. Worse still, even though this solution converges in some cases, it cannot ensure that the final policy is stronger than the initial policy because of the non-transitivity [71]. Specifically, given π_t^v beats π_{t-1}^α, π_t^α beats π_t^v and π_{t+1}^v beats π_t^α, we cannot derive neither π_{t+1}^v could beat π_{t-1}^α nor π_{t+1}^v is stronger than π_t^v. Similar, we also cannot derive that π_t^α is stronger than π_{t-1}^α. As such, we cannot conclude that iterative adversarial retraining could keep searching for policies close to the NE point.

A.2 Proof of Lemma 1

Lemma 1. *Given an approximation V_η^1 of V^1 obtained from $M \geq \frac{C}{\epsilon^2} log \frac{2}{p}$ episodes, we have $Pr(|V^1 - V_\eta^1| \leq \epsilon) \geq 1 - p$.*

Proof of Lemma 1. As is mentioned in Section 4.3, because the ground-truth V^1 is unknown, we first utilize the Monte Carlo approximation to approximate V^1 and obtain $\tilde{V}^1$. Then, we learn a DNN V_η^1 to fit $\tilde{V}^1$ via Eqn. (7). By the bounded reward assumption (*i.e.,* the reward is within a range $[r_0, r_1]$) and Hoeffding's inequality [77], we have

$$Pr(|V^1 - \tilde{V}^1| \leq \epsilon) \geq 1 - 2e^{-\epsilon^2 M/C}, \tag{9}$$

where C is a constant. As such, if $M \geq \frac{C}{\epsilon^2} log \frac{2}{p}$, we have

$$Pr(|V^1 - \tilde{V}^1| \leq \epsilon) \geq 1 - 2e^{-2\epsilon^2 M/C} \geq 1 - p\,. \tag{10}$$

According to the universal approximation theorem of DNN [79], when approximating a target function in any form, by selecting the DNN with enough capacity, one could guarantee that the approximation error can be arbitrarily low. In our case, we can find a proper DNN model to approximate $\tilde{V}^1$ with an approximation error that is lower than ϵ. As such, we have the following inequality $\max |\tilde{V}^1 - V_\eta^1| < \epsilon_1$, where $\epsilon_1 \leq \epsilon$. By combining this inequality with Eqn. (10), we have $Pr(|V^1 - V_\eta^1| \leq \epsilon) \geq 1 - p$. □

A.3 Proof of Theorem 2.

First, we prove that with a bounded learning rate, each pair of policy (π_k^1, π_k^2) could guarantee converge to the NE point, when using the true value function. For the notation simplicity, we drop k and use one policy pair for the proof. Using the PPO algorithm to update π_i^1 against π_v^2 (the strongest opponent in the rest $K-1$ policy pair), we have the following inequality.

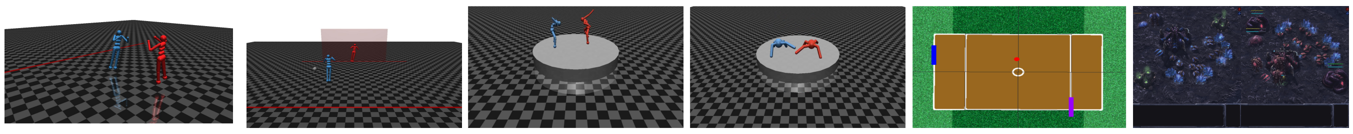
$$|V^1(\pi_{i+1}^1, \pi_v^2) - L_{\pi_i^1}(\pi_{i+1}^1, \pi_v^2)| \leq C1(\max_s \mathbb{TV}(\pi_i^1(\cdot|s)||\pi_{i+1}^1(\cdot|s)))^2\,, \tag{11}$$

where $L_{\pi_i^1}(\pi_{i+1}^1, \pi_v^2) = V^1(\pi_i^1, \pi_v^2) + C2$ and $C1 = \frac{4\epsilon\gamma}{(1-\gamma)^2} > 0$, where i represents the ith iteration. The proof of this inequality can be found in [63]. Substituting $L_{\pi_i^1}(\pi_{i+1}^1, \pi_v^2)$ into the Eqn. (11) and apply the absolute value inequality $||a| - |b|| \leq |a - b|$, we get

$$\begin{aligned} &-C1(\max_s \mathbb{TV}(\pi_i^1(\cdot|s)||\pi_{i+1}^1(\cdot|s)))^2 + |C2| \leq (V^1(\pi_{i+1}^1, \pi_v^2) - V^1(\pi_i^1, \pi_v^2)) \\ &\leq C1(\max_s \mathbb{TV}(\pi_i^1(\cdot|s)||\pi_{i+1}^1(\cdot|s)))^2 + |C2|\,, \end{aligned} \tag{12}$$

from this inequality, we also have

$$-C1||\pi_i^1 - \pi_{i+1}^1||^2 + |C2| \leq C1||\pi_i^1 - \pi_{i+1}^1||^2 + |C2|\,. \tag{13}$$



(a) You-Shall-Not-Pass. (b) Kick-And-Defend. (c) Sumo-Humans. (d) Sumo-Ants. (e) Pong. (f) StarCraft II.

Figure 5: The screenshot of the selected environments. In each game the blue agent is the player_1 and the other is the player_2.

Similarly, updating π_i^2 against π_u^1 (the strongest opponent to π_i^2) with the PPO algorithm gives

$$-C3||\pi_i^2 - \pi_{i+1}^2||^2 + |C4| \leq C3||\pi_i^2 - \pi_{i+1}^2||^2 + |C4|\,. \tag{14}$$

Denote g_i^1 and g_i^2 as the of V^1 w.r.t. π_i^1 and π_2^2, we have $\pi_{i+1}^1 - \pi_i^1 = \eta_i g_i^1$ and $\pi_{i+1}^2 - \pi_i^2 = -\eta_i g_i^2$, where η_i is the learning rate at iteration i. If η_i is bounded by

$$\begin{aligned} \eta_i||g_i^1||^2 &\leq -C1\eta_i^2||g_i^1||^2 + |C2|\,, \\ \eta_i||g_i^2||^2 &\geq C3\eta_i^2||g_i^2||^2 + |C4|\,, \end{aligned} \tag{15}$$

we can obtain the following inequality

$$\begin{aligned} &< g_i^1, \pi_{i+1}^1 - \pi_i^1 > \leq V^1(\pi_{i+1}^1, \pi_v^2) - V^1(\pi_i^1, \pi_v^2)\,, \\ &< g_i^2, \pi_{i+1}^2 - \pi_i^2 > \geq V^1(\pi_u^i, \pi_{i+1}^2) - V^1(\pi_u^i, \pi_i^2)\,. \end{aligned} \tag{16}$$

With the inequality in Eqn. (16), we then prove the convergence of (π_i^1, π_i^2) under the condition of Eqn. (15). Specifically, we first prove the distance between the current (π_i^1, π_i^2) and the optimal (π_*^1, π_*^2) keeps decreasing as training iteration i increases. Expanding the squared distance gives

$$||\pi_{i+1}^1 - \pi_*^1||^2 == ||\pi_i^1 - \pi_*^1||^2 + 2\eta_i < g_i^1, \pi_i^1 - \pi_*^1 > + \eta_i^2||g_i^1||^2\,. \tag{17}$$

According to the inequality about π^1 in Eqn. (16), we have

$$< g_i^1, \pi_i^1 - \pi_*^1 > \leq V^1(\pi_i^1, \pi_v^2) - V^1(\pi_*^1, \pi_v^2)\,. \tag{18}$$

Plugging Eqn.(18) into Eqn.(17) gives

$$||\pi_{i+1}^1 - \pi_*^1||^2 \leq ||\pi_i^1 - \pi_*^1||^2 + 2\eta_i(V^1(\pi_i^1, \pi_v^2) - V^1(\pi_*^1, \pi_v^2)) + \eta_i^2||g_i^1||^2\,. \tag{19}$$

Similarly for π_i^2, we have

$$||\pi_{i+1}^2 - \pi_*^2||^2 = ||\pi_i^2 - \pi_*^2||^2 - 2\eta_i < g_i^2, \pi_i^2 - \pi_*^2 > + \eta_i^2||g_i^2||^2\,. \tag{20}$$

According to the inequality about π^2 in Eqn. (16), we have

$$< g_i^2, \pi_i^2 - \pi_*^2 > \geq V^1(\pi_u^1, \pi_i^2) - V^1(\pi_u^1, \pi_*^2)\,. \tag{21}$$

Then, we can also derive

$$||\pi_{i+1}^2 - \pi_*^2||^2 \leq ||\pi_i^2 - \pi_*^2||^2 - 2\eta_i(V^1(\pi_u^1, \pi_i^2) - V^1(\pi_u^1, \pi_*^2)) + \eta_i^2||g_i^2||^2\,. \tag{22}$$

Adding Eqn. (19) with Eqn. (22) gives

$$\begin{aligned} &||\pi_{i+1}^1 - \pi_*^1||^2 + ||\pi_{i+1}^2 - \pi_*^2||^2 \\ \leq &||\pi_i^1 - \pi_*^1||^2 + ||\pi_i^2 - \pi_*^2||^2 - 2\eta_i(V^1(\pi_*^1, \pi_v^2) - V^1(\pi_u^1, \pi_*^2) + \\ &V^1(\pi_u^1, \pi_i^2) - V^1(\pi_i^1, \pi_v^2)) + \eta_i^2(||g_i^1||^2 + ||g_i^2||^2)\,. \end{aligned} \tag{23}$$

Recall the NE condition gives the following inequality

$$V^1(\pi_u^1, \pi_*^2) \leq V^1(\pi_*^1, \pi_*^2) \leq V^1(\pi_*^1, \pi_v^2)\,. \tag{24}$$

Plugging Eqn. 24 into Eqn. (23) gives

$$\begin{aligned} &||\pi_{i+1}^1 - \pi_*^1||^2 + ||\pi_{i+1}^2 - \pi_*^2||^2 \\ \leq &||\pi_i^1 - \pi_*^1||^2 + ||\pi_i^2 - \pi_*^2||^2 - 2\eta_i(V^1(\pi_u^1, \pi_i^2) - V^1(\pi_i^1, \pi_v^2)) + \eta_i^2(||g_i^1||^2 + ||g_i^2||^2)\,. \end{aligned} \tag{25}$$

Let $E_i = V^1(\pi_u^1, \pi_i^2) - V^1(\pi_i^1, \pi_v^2)$. According to Line 5&6 in Algorithm 1, we can also derive $E_i \geq 0$. If the learning rate satisfies $\eta_i < \frac{2E_i}{|g_i^1||^2 + ||g_i^2||^2}$, we have

$$||\pi_{i+1}^1 - \pi_*^1||^2 + ||\pi_{i+1}^2 - \pi_*^2||^2 \leq ||\pi_i^1 - \pi_*^1||^2 + ||\pi_i^2 - \pi_*^2||^2\,. \tag{26}$$

Combining the condition above with (15), we have if the following condition holds

$$\sqrt{\frac{|C4|}{C3||g_i^2||^2}} \leq \eta_i \min\left(\frac{2E_i}{|g_i^1||^2 + ||g_i^2||^2}, \sqrt{\frac{|C2|}{C1||g_i^1||^2} - \frac{2E_i}{C1(||g_i^1||^2 + ||g_i^2||^2)}}\right), \tag{27}$$

the difference between (π_i^1, π_i^2) and (π_*^1, π_*^2) strictly decreases during the training process.

With this property, we then prove that the difference decreases to zero with a reasonable large number of policy pairs K. Suppose the training process converges to an accumulation point (π_T^1, π_T^2), such that its difference with the NE policy no longer decreases. At this point, we have $E_T = 0$. This is because, if $E_T > 0$, one could always find a proper learning rate η_T such that $(\pi_{T+1}^1, \pi_{T+1}^2)$ has a lower distance to the NE policy than (π_T^1, π_T^2), indicating the training has not converged yet. As such, when training converges, we have $E_T = V^1(\pi_u^1, \pi_T^2) - V^1(\pi_T^1, \pi_v^2) = 0$. With a reasonable large K, we have π_v^2 and π_u^1 equals to $\text{argmin}_{\pi^2}(V^1(\pi_T^1, \pi^2))$ and $\text{argmax}_{\pi^1}(V^1(\pi^1, \pi_T^2))$. This gives the following equality

$$V^1(\pi_T^1, \text{argmin}_{\pi^2}(V^1(\pi_T^1, \pi^2))) = V^1(\text{argmax}_{\pi^1}(V^1(\pi^1, \pi_T^2)), \pi_T^2)\,. \tag{28}$$

We also have the following inequality

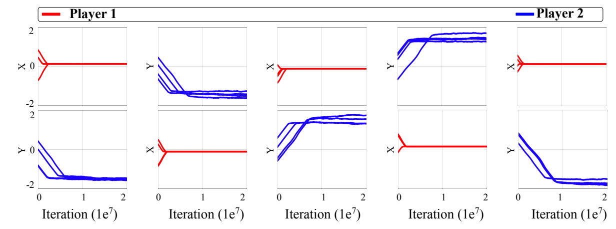
$$V^1(\pi_T^1, \text{argmin}_{\pi^2}(V^1(\pi_T^1, \pi^2))) \leq V^1(\pi_T^1, \pi_T^2) \leq V^1(\text{argmax}_{\pi^1}(V^1(\pi^1, \pi_T^2)), \pi_T^2)\,. \tag{29}$$

Combine Eqn. (29) and Eqn. (28), we can derive

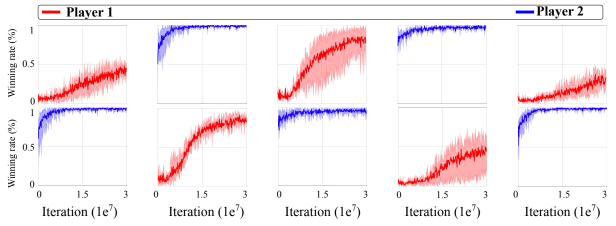
$$V^1(\pi_T^1, \text{argmin}_{\pi^2}(V^1(\pi_T^1, \pi^2))) = V^1(\pi_T^1, \pi_T^2) = V^1(\text{argmax}_{\pi^1}(V^1(\pi^1, \pi_T^2)), \pi_T^2)\,. \tag{30}$$

This equation shows the minimax value and maximin value are the same at (π_T^1, π_T^2). It also gives $V^1(\pi^1, \pi_T^2) \leq V^1(\pi_T^1, \pi_T^2) \leq V^1(\pi_T^1, \pi^2)$ for any π^1 and π^2. As a result, (π_T^1, π_T^2) is the NE policy of V^1.

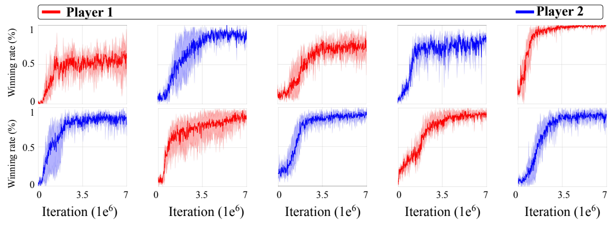
In real-world RL tasks, the ground-truth value function is unknown. As specified in Section 4.3, we use a neural network V_η^1 to approximate the V^1. According to Lemma 1, the approximation error is bounded by ϵ. Plugging Lemma 1, we have $V^1(\pi^1, \pi_T^2) - \epsilon \leq V^1(\pi_T^1, \pi_T^2) \leq V^1(\pi_T^1, \pi^2) + \epsilon$, for all π^1 and π^2. With a bounded learning rate, our algorithm is guaranteed to converge to a ϵ-approximate NE point. □



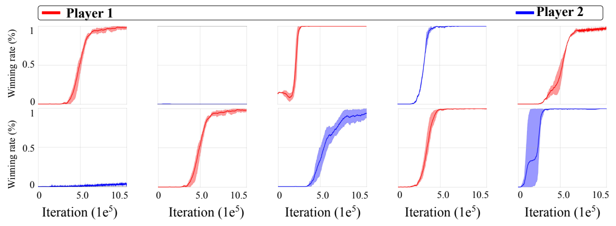
(a) ④-S: Euclidean game with non-convex non-concave value function.



(b) You-Shall-Not-Pass.



(c) Pong.



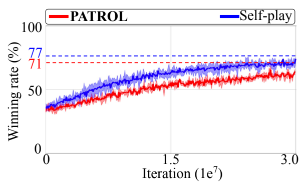
(d) StarCraft II.

Figure 6: Iteratively adversarial retraining results of 10 rounds. In each game, we start with using player_1 as the adversarial agent and player_2 as the victim agent.

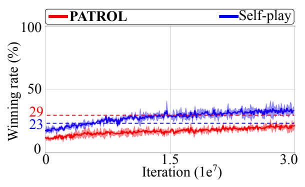
B Additional Experiments

B.1 Additional Experiment Setup

Hyper-parameters. For the hyper-parameters shared by PATROL and the baselines, we use the same set of choices. We follow [25], [55], and [69] to set hyper-parameters. For the You-Shall-Not-Pass game, we use a multi-layer perceptron with the architecture of MLP-380-64-64-17. It has four layers and the numbers indicate the number of neurons in each layer. The architecture of the other games are: MLP-380-128-LSTM-128-MLP-128-17 for the Kick-And-Defend game, MLP-395-128-LSTM-128-MLP-128-17 for the Sumo-Humans game, MLP-137-128-LSTM-128-MLP-128-8 for the Sumo-Ants game, MLP-13-64-64-2 for the Pong game, and MLP-754-128-128-128-165 for the StarCraft II game. For



(a) MuJoCo Player 1 as victim.



(b) MuJoCo Player 2 as victim.

Figure 7: The robustness of PATROL and Self-play against the attack in [27] in the You-Shall-Not-Pass game.

toy games, we directly learn the policy without using a neural network. We use the ADAM optimizer with a learning rate of 0.00001 for the StarCraft II and 0.0001 for all the others.

B.2 Runtime & Hyper-parameter Sensitivity

Runtime. To compare the computational cost of PATROL and Self-play, we record the training time of both methods on the MuJoCo games using the same machine (a server with 2 AMD EPYC 7702 64-Core CPUs and 4 NVIDIA RTX A6000 GPUs). The average runtime to the convergence of PATROL is about 1.8× over the self-play baseline (41.5h vs. 24h on You-Shall-Not-Pass and 30h vs. 18h on Sumo-Humans).
Sensitivity. To test the sensitivity of PATROL to the number of policy pairs K, we train three policies with $K = 1/2/3$ for each player in the You-Shall-Not-Pass game, play them against the same self-play policy, and record the winning rates: PATROL_1 vs. Self-play_2: 33.0% ($K = 1$), 42.0% ($K = 2$), 44.0% ($K = 3$); PATROL_2 vs. Self-play_1: 81.0% ($K = 1$), 86.0% ($K = 2$), 86.0% ($K = 3$). $K = 2/3$ gives stronger policies than $K = 1$, verifying the efficacy of training multiple policy pairs. Besides, K equals 2 and 3 give similar results, confirming our statement in Section 5.1.

B.3 Iteratively Adversarial Retraining

We run the iteratively adversarial retraining on four games and show the results in 6. For the toy game, we report the trained policy (*i.e.*, value of X and Y), and for other games, we show the changes in the adversarial winning rate. As shown in the figure, iteratively adversarial retraining fails to converge to a fixed pair of policies on all games.

B.4 PATROL against a New Attack

Recently, [27] proposed a new adversarial attack for two-player competitive games that are not exactly zero-sum. Here, we turn the You-Shall-Not-Pass game into such a case by adding intermediate rewards and train the normal policies with PATROL and Self-play. Then, we use the method in [27] to attack these policies and show the results in Fig. 7. The result is aligned with those in Fig. 3&4, confirming the robustness of our method against this new attack and further verifying our robustness guarantee.