Adversarial Policy Learning in Two-player Competitive Games

Wenbo Guo [1] **Xian Wu** [1] **Sui Huang** [2] **Xinyu Xing** [1]

Abstract

In a two-player deep reinforcement learning task, recent work shows an attacker could learn an adversarial policy that triggers a target agent to perform poorly and even react in an undesired way. However, its efficacy heavily relies upon the zero-sum assumption made in the two-player game. In this work, we propose a new adversarial learning algorithm. It addresses the problem by resetting the optimization goal in the learning process and designing a new surrogate optimization function. Our experiments show that our method significantly improves adversarial agents' exploitability compared with the state-of-art attack. Besides, we also discover that our method could augment an agent with the ability to abuse the target game's unfairness. Finally, we show that agents adversarially retrained against our adversarial agents could obtain stronger adversary-resistance.

1. Introduction

Along with the great success of Deep Reinforcement Learning (DRL) in the application of two-player games comes a new form of attack that exploits the weakness of a policy network to fail the corresponding game agent. In the past, the development of this new kind of attack makes many unrealistic assumptions. The most common one is that an attacker has the capability of manipulating the observation of an agent by altering the environment with which the agent interacts. Under this assumption, researchers have proposed a variety of technical methods, demonstrating significant effectiveness in failing a game agent in a complicated game task (*e.g.,* (Huang et al., 2017; Lin et al., 2020)).

Sharing the similar viewpoint elaborated in (Gleave et al., 2020), we argue that attacks developed under this assumption are not practical. For example, given a master agent deployed in an online video game platform playing with hundreds of thousands of professional participants, the manipulation of the agent's observation could mean the free alternation of pixel values in the game environment (*e.g.,* changing the canvas color of the Atari game (ATARI, 2006)). In this context, to achieve this objective, an attacker might have to spend thousands of hours of effort on hacking the game platform and expect to obtain unauthorized access to the master agent and the environment with which it interacts. Given the advance of software hardening technology, such an expectation cannot always be guaranteed.

In this work, we do not bake in this assumption and propose a new attack in the context of a two-player competitive game. Like the recent research (Gleave et al., 2020), we train an adversarial agent that interacts with a well-trained victim agent in a two-player game environment and triggers it to react in an undesired fashion. Differently, as we will elaborate in Section 3.1, our learning method, however, relaxes an implicit and sometimes unrealistic assumption – a two-player game must strictly follow a zero-sum setting. We show this relaxation makes adversarial learning effective for more sophisticated games (*e.g.,* StarCraft). Besides, our learning method goes beyond exploiting the weakness of a victim agent. It demonstrates an ability to abuse the target game's unfairness when an adversarial agent has minimal impact on the victim's decision making.

Technically, instead of the straightforward adoption of an existing RL approach as is proposed in (Gleave et al., 2020), our adversarial learning method introduces a new learning algorithm that not only maximizes the expected reward of the adversarial agent but, more importantly, minimizes that of the victim. As is specified in Section 3.2, to build our learning algorithm correctly, we have to tackle a monotonicity challenge or, in other words, ensure the update of an adversarial policy does not incur the random fluctuation of the learning objective. To do this, we first remodel a two-player game and define both the adversary and victim's expected rewards as a function of the adversarial policy. Second, we carefully design a new objective function that could theoretically guarantee the monotonic property in the entire policy learning process.

[1]College of Information Sciences and Technology, The Pennsylvania State University, State College, PA, USA [2]Netflix Inc., Los Gatos, CA, USA. Correspondence to: Wenbo Guo <wzg13@ist.psu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

With all these technical endeavors above, Section 4 shows that we bring fundamental advantages over the state-of-the-art technique (Gleave et al., 2020) from many aspects. First, we can enable effective, practical adversarial attacks against sophisticated games (*e.g.,* StarCraft). To the best of our knowledge, this is the first work that successfully demonstrates a practical attack against a sophisticated video game (*i.e.,* StarCraft II). Second, we can exploit the unfairness of a game to defeat a target victim agent even if the adversarial agent has minimal influence on the victim's decision making. Third, we can learn an adversarial agent with stronger exploitability and better transferability. It makes the attack more powerful and practical. Fourth, our learning algorithm overall demonstrates less performance variance in the agent learning process than the existing method. It indicates that, by using our approach for learning an adversarial policy, an attacker could better receive an agent with a consistent hostile capability. Last but not least, our adversarial agent can help agent developers learn an agent with stronger adversary-resistance. We released our source code to support future research.[1]

2. Related Work

Existing attacks on DRL policies primarily follow three methods – ❶ attack through observation manipulation, ❷ attack through action/trajectory manipulation, and ❸ attack through an adversarial agent. Below, we summarize these works and discuss their differences from ours.

Attack through observation manipulation. Following the conventional attacks on DNNs (Goodfellow et al., 2015; Papernot et al., 2016; Carlini & Wagner, 2017; Madry et al., 2018), Huang et al. (2017) and Behzadan & Munir (2017) first proposed to perturb the victim's observation at each time step, forced its policy network to output sub-optimal actions, and thus failed the corresponding task. As a follow-up, recent efforts (Kos & Song, 2017; Russo & Proutiere, 2019; Lin et al., 2017; Sun et al., 2020; Zhang et al., 2021) improved the efficiency of such attacks by manipulating the observation of a victim agent at some selected time steps rather than the entire training trajectories.

In addition to the efficiency improvement, some existing works (Huang et al., 2017; Behzadan & Munir, 2017; Zhao et al., 2019; Xiao et al., 2019; Lin et al., 2020) extended the observation manipulation attacks to black-box settings, in which attacks can access only the input and output of a victim agent's policy network or those of a Q network. In this work, we do not assume the full privilege of manipulating an agent's observations but only the control over an adversarial agent. As is discussed in Gleave et al. (2020), this relaxation makes the attack more realistic and cost-effective.

Attack through action/trajectory manipulation. Different from the observation manipulation attacks mentioned above, another line of research directly perturbs the output of the policy network or, in other words, manipulates the action taken by victim agents. Similar to the observation manipulation attacks, researchers have also demonstrated the effectiveness of action manipulation in both black-box and white-box settings (Lee et al., 2020; Xiao et al., 2019).

Inspired by the observation and action manipulation attacks launched at the testing phase, some other researchers also proposed to launch an attack at the training phase. Specifically, they demonstrated that, by manipulating the reward in training trajectories, attackers could train a victim agent failing its corresponding task (Ma et al., 2019; Lykouris et al., 2019; Yang et al., 2019; Kiourti et al., 2019). Like the statement made above, our work removes the assumption of providing an adversary with the ability to vary action or trajectories, making the attack more practical.

Attack through an adversarial agent. Unlike the two attack methods mentioned above, there is an emerging attack, focusing on training an adversarial agent to beat its opponent in a two-player game. For example, Gleave *et al.* (Gleave et al., 2020) trained an adversarial agent for a set of MuJoCo games (Todorov et al., 2012) by using the PPO algorithm. They have demonstrated that their adversarial agent could defeat its opponent agent, exhibiting a higher winning rate. In this work, our method follows an entirely different idea. It can theoretically guarantee the adversarial agent could better discover and exploit its opponent's weakness even if a two-player game does not strictly follow a zero-sum setting.

3. Proposed Technique

3.1. Problem Scope and Assumption

In this work, we fix one agent's policy (either deterministic or stochastic policy), and train the other to win the corresponding two-player Markov game. Note that this setup is the same as the one in (Gleave et al., 2020). It simulates a real-world scenario. A game vendor first releases a master RL agent. Then, an attacker trains an adversarial agent to exploit this master's weakness and win the game for fun or profit (Supplement S6 shows an initial experiment of attacking a victim that varies its policy). Under this setup, our work makes the following two assumptions. First, we relax an implicit assumption of the existing attack (Gleave et al., 2020). That is, increasing one player's reward will decrease the other player's reward gain. This is true for all zero-sum two player games. However, in most of the real-world

[1]https://github.com/psuwuxian/rl_adv_valuediff

two-player games, the rewards are not designed as exactly zero-sum. One common practice of breaking the zero-sum equilibrium is to assign both agents with the same positive or negative reward when a draw occurs (Bansal et al., 2018; OpenAI, 2017). The other is to introduce intermediate rewards and bring other benefits to agents. For example, it could help agents establish natural behaviors (Bansal et al., 2018), encourage their correct movements (Unity, 2020; Sun et al., 2018), and punish their actions that break the game rules (Unity, 2020; ATARI, 2006).

Second, following the setup in (Gleave et al., 2020), we assume an adversary can only play his/her adversarial agent with the victim agent in the corresponding game environment but does not has access to the victim player's observation, value function, and policy network. To enable a stronger attack, we further assume that the attacker knows the victim's instant reward. We believe this is a reasonable and practical assumption. In a two-player game, the instant reward of each agent is determined based on the outcome of each round of the game (*e.g.,* win or loss). Besides, it is relevant to some statistics during the play (*e.g.,* the number of enemies killed and the number of collected resources in StarCraft II (Sun et al., 2018)). This information is always available for each player of the game. In Supplementary Section S7, we demonstrate the transferability of our attack. It implies that, even if instant rewards are not accessible, an attacker could still train an adversarial agent in an environment with visible instant rewards and then successfully launch attacks against the victims.

3.2. Technical Overview and Challenge

As is mentioned above, the existing attack (Gleave et al., 2020) relies on the zero-sum assumption, which implies the increase in one player's reward gain will result in the decrease in the other player's gain. Based on this assumption, by only maximizing the adversarial player's expected total reward using the PPO algorithm, this attack could search for an adversarial policy that consistently decreases the victim reward and thus fails the victim agent. However, for most of the real-world Markov games, where the rewards are not zero-sum, merely maximizing the adversarial reward cannot guarantee to reduce that of the victim agent. As we will show later in Section 4, without such guarantee, the existing attack cannot effectively explore the victim's weakness nor beating the victim. As such, unlike the state-of-the-art attack (Gleave et al., 2020), our new approach does not learn an adversarial agent by simply maximizing the rewards of the adversary through the PPO algorithm. Rather, it searches adversarial policies by providing the adversarial agent with the ability to not only maximize its rewards but, more importantly, impose negative influence upon the victim agent. For example, in the StarCraft game, this could be viewed as a learning strategy that, on the one hand, builds up a strong economy and army to countervail its opponent and, on the other hand, intervenes in the movements of the opponent and thus limits its expected rewards. Mathematically, this can be viewed as

$$J(\theta) = \text{maximize}(V_{\pi}^{\alpha}(s) - V_{\pi}^{\upsilon}(s))\,, \tag{1}$$

which maximizes the value function $V_{\pi}^{\alpha}(s)$ of the adversarial agent and meanwhile minimizes that of the victim agent. Here, $\pi = (\pi^{\alpha}, \pi^{\upsilon})$ is a joint policy.

While the idea mentioned above is straightforward and intuitive, using Eqn. (1) as the objective function for learning an adversarial agent still confronts two critical challenges. First, since we do not have the victim observation, we cannot directly approximate the victim value function $V_{\pi}^{\upsilon}(s)$ via a neural network that takes its observation as input. To solve this problem, we remodel this adversarial agent learning problem, transform the two-player Markov game into a one-player Markov Decision Process (MDP). As we will specify in the follow-up session, with our remodeling, the victim value function can be redefined as $V_{\pi^{\alpha}}^{\upsilon}(s)$, the output of which depends only upon the information pertaining to the adversary.

After reformalizing the victim value function, a trivial solution for resolving Eqn. (1) is to apply the vanilla policy gradient method (Konda & Tsitsiklis, 2000) and approximate both value functions with two individual neural networks (*i.e.,* $G_{\pi^{\alpha}}^{\alpha}(s)$ and $G_{\pi^{\alpha}}^{\upsilon}(s)$). However, due to the limitation rooted in the vanilla policy gradient method, such a solution cannot guarantee the monotonic changes in both value functions. As a result, a more advanced solution is to replace the $V_{\pi}^{\alpha}(s)$ in Eqn. (1) with the surrogate objective proposed in the TRPO algorithm (Schulman et al., 2015), denoted as $M_{\pi^{\alpha}}^{\alpha}(s)$. Intuitively, this surrogate objective (*i.e.,* $M_{\pi^{\alpha}}^{\alpha}(\cdot) - G_{\pi^{\alpha}}^{\upsilon}(\cdot)$) could ensure the monotonic increase of the adversarial reward. However, it cannot provide the monotonic property for the victim player. In addition, it is also unclear whether the newly added second term will break the monotonic property for the first term.[2]

Inspired by the design of TRPO, we tackle the monotonicity challenge above by designing an approximation for the expected reward difference $V_{\pi^{\alpha}}^{\alpha}(s) - V_{\pi^{\alpha}}^{\upsilon}(s)$. As we will present and prove in the follow-up section, this approximation can ensure the monotonic property for the value difference (*i.e.,* $V_{\pi_{old}^{\alpha}}^{\alpha}(s) - V_{\pi_{old}^{\alpha}}^{\upsilon}(s) \leq V_{\pi_{new}^{\alpha}}^{\alpha}(s) - V_{\pi_{new}^{\alpha}}^{\upsilon}(s)$).

3.3. Technical Details

Remodeling two-player Markov Game. A two-player Markov game (Zhang et al., 2019) can be represented as

[2] As we show in Supplementary Section S6, without ensuring the monotonic property, this more advanced method cannot train powerful adversarial agents.

$(\mathcal{N}, \mathcal{S}, \{\mathcal{A}\}^i_{i\in\mathcal{N}}, \mathcal{P}, \{R^i\}_{i\in\mathcal{N}}, \gamma)$. $\mathcal{N} = \{\alpha, \upsilon\}$ denotes the agent set, in which α and υ stand for the adversary and victim, respectively. $\mathcal{S}$ denotes the state space observed by both agents, $\mathcal{A}^i$ represents the action space of agent i. $\mathcal{P} : \mathcal{S} \times \mathcal{A}^\alpha \times \mathcal{A}^\upsilon \to \Delta(\mathcal{S})$ denotes the transition probability for any joint actions of both agents. R^i: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for agent i. γ is the discount factor. The state-value function for each agent is defined as a function of the joint policy $\pi(a|s) = \prod_{i\in\mathcal{N}} \pi^i(a^i|s)$

$$V^i_\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[R^i(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}}[V^i_\pi(s')]] \,. \tag{2}$$

As is mentioned in Section 3.1, the victim agent follows a fixed policy. Under this setup, we have the following proposition (see the proof in Supplementary Section S1).

Proposition 1. *In a two-player Markov game, if one agent follows a fixed policy, the state transition of the game system will depend only upon the policy of the other agent rather than their joint policy.*

With the proposition above, we can redefine both agents' state-value and action-value functions as the functions of the adversarial policy below.

$$\begin{aligned} Q^i_{\pi^\alpha}(s,a^\alpha) &= R^i(s,a) + \mathbb{E}_{s'\sim P(s'|s,a^\alpha)}[\gamma V^i_{\pi^\alpha}(s')] \,, \\ V^i_{\pi^\alpha}(s) &= \mathbb{E}_{a^\alpha\sim\pi^\alpha}[Q^i_{\pi^\alpha}(s,a^\alpha)] \,, \end{aligned} \tag{3}$$

where $a = (a^\alpha, F^{\pi^\upsilon}(s))$. $F^{\pi^\upsilon}(s)$ represent the actions sampled from the fixed victim policy.

As is shown above, the new state-value and action-value functions no longer enclose the policy of the victim nor its observations or actions. It perfectly addresses the concern of having to know the victim agent.

Constructing and Solving Objective Function. With the new definition above, the objective function shown in Eqn. (1) can be reformalized as

$$J(\theta) = \text{maximize}_\theta(V^\alpha_{\pi^\alpha_\theta}(s) - V^\upsilon_{\pi^\alpha_\theta}(s)). \tag{4}$$

Here, π^α_θ refers to as the adversarial policy network parameterized by θ. As is discussed in Section 3.2, trivial extending neither the vanilla policy gradient method nor the TRPO algorithm could guarantee the monotonic property. To address this problem, we propose the following method.

Based on the Theorem 1 in Schulman et al. (2015), we can approximate $V^\alpha_{\pi^\alpha}(s)$ with $M^\alpha_{\pi^\alpha}(\pi^{\alpha'})$ and thus obtain the following relationship

$$\begin{aligned} V^\alpha_{\pi^{\alpha'}}(s) &\geq M^\alpha_{\pi^\alpha}(\pi^{\alpha'}) \,, \\ M^\alpha_{\pi^\alpha}(\pi^{\alpha'}) &= L^\alpha_{\pi^\alpha}(\pi^{\alpha'}) - C_1\mathbb{KL}^{\max}(\pi^\alpha(\cdot|s)||\pi^{\alpha'}(\cdot|s)) \,, \end{aligned} \tag{5}$$

where C_1 is a constant. $\mathbb{KL}^{\max}(\pi^\alpha(\cdot|s)||\pi^{\alpha'}(\cdot|s)) = \max_s \mathbb{KL}(\pi^\alpha(\cdot|s)||\pi^{\alpha'}(\cdot|s))$. $\pi^{\alpha'}$ and π^α refers to as the new and old adversarial policy, respectively.

To obtain an approximation for $V^\upsilon_{\pi^\alpha}(s)$, we follow the idea in Schulman et al. (2015) and propose the method below. First, we compute the difference of the victim's state-value function before and after the update of the adversarial policy and come up with the following lemma (see the proof in Supplementary Section S2).

Lemma 1. *Given an old policy π^α and a new policy $\pi^{\alpha'}$ in a two-agent Markov game, the difference of the victim state-value function under each policy is as follows.*

$$V^\upsilon_{\pi^{\alpha'}}(s) - V^\upsilon_{\pi^\alpha}(s) = E_{\tau\sim\pi^{\alpha'}}[\sum_{t=0}^{\infty}\gamma^t A^\upsilon_{\pi^\alpha}(s_t, a^\alpha_t)] \,. \tag{6}$$

Intuitively, Lemma 1 indicates that, if the adversarial agent switches to a new policy $\pi^{\alpha'}$ from the old one π^α at the state s, the expectation of the victim's future reward will change. Technically, one can compute the change by applying the victim's advantage function under the old adversarial policy to the trajectories collected by using the new adversarial policy.

To get rid of the summation over infinite time in Eqn. (6), we can further rewrite this equation by taking the summation over all possible states.

$$V^\upsilon_{\pi^{\alpha'}}(s) - V^\upsilon_{\pi^\alpha}(s) = \sum_s \rho_{\pi^{\alpha'}}(s)\sum_a \pi^{\alpha'}(a^\alpha|s)A^\upsilon_{\pi^\alpha}(s,a^\alpha), \tag{7}$$

where $\rho_{\pi^{\alpha'}}(s) = \sum_t \gamma^t P(s_t = s|\pi^{\alpha'})$.

With the rewritten equation in hand, the computation of Eqn. (7) is still hard due to the unknown visitation frequencies of $\pi^{\alpha'}$. To solve this problem, we replace $\rho_{\pi^{\alpha'}}(s)$ with $\rho_{\pi^\alpha}(s)$ and thus approximate $V^\upsilon_{\pi^{\alpha'}}(s)$ with $L^\upsilon_{\pi^\alpha}(\pi^{\alpha'}) = V^\upsilon_{\pi^\alpha}(s) + \sum_s \rho_{\pi^\alpha}(s)\sum_a \pi^{\alpha'}(a^\alpha|s)A^\upsilon_{\pi^\alpha}(s,a^\alpha)$. The relationship between our approximation and the actual value function is described in the theorem below (see the proof in Supplementary Section S3).

Theorem 1. *The difference between $V^\upsilon_{\pi^{\alpha'}}(s)$ and $L^\upsilon_{\pi^\alpha}(\pi^{\alpha'})$ is bounded by:*

$$\begin{aligned} V^\upsilon_{\pi^{\alpha'}}(s) &\leq L^\upsilon_{\pi^\alpha}(\pi^{\alpha'}) + C_2\mathbb{KL}^{\max}(\pi^\alpha||\pi^{\alpha'}) = M^\upsilon_{\pi^\alpha}(\pi^{\alpha'}) \,, \\ C_2 &= \frac{4\max_{s,a^\alpha}|A^\upsilon_{\pi^\alpha}(s,a^\alpha)|\gamma}{(1-\gamma)^2} \,. \end{aligned} \tag{8}$$

Using the inequality in the theorem above along with that in Eqn. (5), we can easily derive the following inequality

$$V^\alpha_{\pi^{\alpha'}}(s) - V^\upsilon_{\pi^{\alpha'}}(s) \geq M^\alpha_{\pi^\alpha}(\pi^{\alpha'}) - M^\upsilon_{\pi^\alpha}(\pi^{\alpha'}) = M_{\pi^\alpha}(\pi^{\alpha'}) \,. \tag{9}$$

Further, we can derive

$$M_{\pi^\alpha}(\pi^\alpha) = L^\alpha_{\pi^\alpha}(\pi^\alpha) - L^\upsilon_{\pi^\alpha}(\pi^\alpha) = V^\alpha_{\pi^\alpha} - V^\upsilon_{\pi^\alpha} \,. \tag{10}$$

Then, by maximizing $M_{\pi^\alpha}(\pi^{\alpha'})$ via gradient ascent at each iteration, we can ensure $M_{\pi^\alpha}(\pi^\alpha) \leq M_{\pi^\alpha}(\pi^{\alpha'})$

and thus guarantee the monotonic property for Eqn. (4) (*i.e.,* $(V^{\alpha}_{\pi^{\alpha}}(s) - V^{\upsilon}_{\pi^{\alpha}}(s)) \leq (V^{\alpha}_{\pi^{\alpha'}}(s) - V^{\upsilon}_{\pi^{\alpha'}}(s))$). Specifically, $M_{\pi^{\alpha}}(\pi^{\alpha'})$ equals to the following Equation

$$M_{\pi^{\alpha}}(\pi^{\alpha'}) = \sum_s \rho_{\pi^{\alpha}}(s) \sum_a \pi^{\alpha'}(a^{\alpha}|s)(A^{\alpha}_{\pi^{\alpha}}(s,a^{\alpha}) - A^{\upsilon}_{\pi^{\alpha}}(s,a^{\alpha})) - C\mathbb{KL}^{\max}(\pi^{\alpha}||\pi^{\alpha'}) + C_3 \,, \tag{11}$$

where $C = C_1 - C_2$ and $C_3 = (V^{\alpha}_{\pi^{\alpha}}(s) - V^{\upsilon}_{\pi^{\alpha}}(s))$ are constants. To solve this Eqn. (11), we first follow the TRPO algorithm and transform the $C\mathbb{KL}^{\max}(\pi^{\alpha}||\pi^{\alpha'})$ into a trust region constraint over the average KL-divergence. Even after transformation, the optimization is still hard to implement due to the summation over the new policy $\pi^{\alpha'}$. To solve this problem, we apply importance sampling and replace it with the summation over the old policy that can be computed via the Monte Carlo method (Thrun, 2000). Then, we further replace the trust region constrains with clipped ratio operation introduced in the PPO algorithm and obtain our final optimization function.

$$\begin{aligned} &\text{argmax}_{\theta}\, \mathbb{E}_{(a^{\alpha}_t, s_t)\sim\pi^{\alpha}_{old}}[\min(\text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A^{\alpha}_t, \rho_t A^{\alpha}_t) \\ &- \min(\text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A^{\upsilon}_t, \rho_t A^{\upsilon}_t)] \,, \\ &\rho_t = \frac{\pi^{\alpha}_{\theta}(a^{\alpha}_t|s_t)}{\pi^{\alpha}_{old}(a^{\alpha}_t|s_t)}, A^{\alpha}_t = A^{\alpha}_{\pi^{\alpha}_{old}}(a^{\alpha}_t, s_t), A^{\upsilon}_t = A^{\upsilon}_{\pi^{\alpha}_{old}}(a^{\alpha}_t, s_t) \,. \end{aligned} \tag{12}$$

To resolve an adversarial policy, we first approximate the adversarial and victim value function with two neural networks through the TD-Learning (Tesauro, 1995) and then update the adversarial policy network by optimizing Eqn. (12). For a more detailed description of how to obtain Eqn. (12) and our learning algorithm, readers could refer to the Supplementary Section S4.

4. Evaluation

In this section, we evaluate our proposed learning algorithm by using five selected games (*i.e.,* four MuJoCo games and StarCraft II). Due to space limit, we specify the implementation details and experiment setup (*i.e.,* game and victim policy selection, evaluation metric, hyperparameters) in Supplementary Section S5.

4.1. Exploitability

Experiment Design. In our first experiment, we use the state-of-the-art approach (Gleave et al., 2020) as our baseline and compare it with our proposed attack. To be specific, given a two-player game as well as a victim agent well-trained for that game, we train our adversarial agent against the victim by using the proposed learning algorithm and compare its winning rate with that collected from the state-of-the-art method.

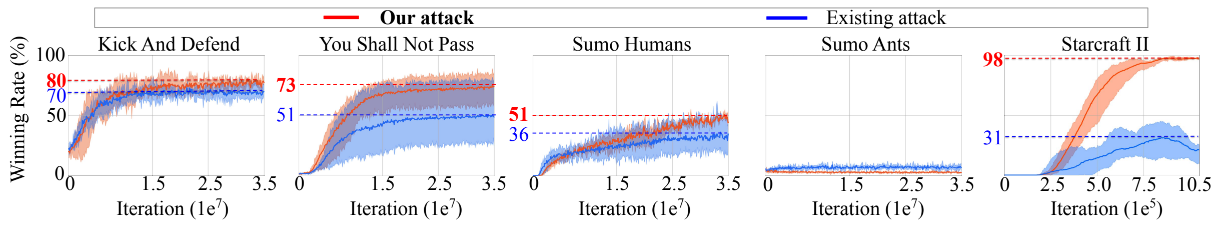
Experiment Results. Figure 1 shows the winning rate comparison between our adversarial agents and that obtained by the existing attack. As we can observe first, overall, the adversarial agent trained by our proposed method demonstrates higher winning rates (or in other words, stronger exploitability) than that prepared by the state-of-the-art technique (Gleave et al., 2020). This result confirms that by maximizing the total reward difference between the adversary and victim, our method possesses more potential to discover an effective adversarial policy and thus demonstrates higher winning rates than the existing attack that only maximizes the adversarial expected total reward.

Figure 1(a) also shows that our method demonstrates a more significant increase in the winning rate on StarCraft II game (98% vs. 31%) than MuJoCo games. It is because for MuJoCo games, the increase in the adversarial agent's reward, to some extent, contributes to the decrease in the victim's reward gain. However, the StarCraft II has a more sophisticated intermediate reward system, in which the increase in adversarial reward has minimal impact on the victim reward gain.

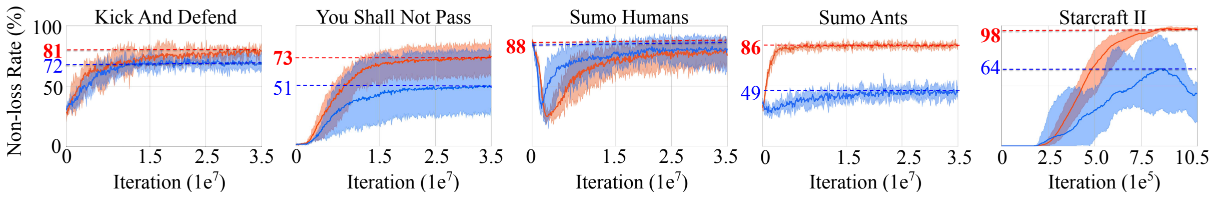
In addition to the adversarial winning rate, we also compare the behaviors of the adversarial agents learned through our proposed method and that learned through the baseline approach. We illustrate the agent behaviors through demo videos at `https://tinyurl.com/y3ax4ayk`. On the three MuJoCo games (*i.e.,* You-Shall-Not-Pass, Kick-And-Defend, and Sumo-Humans), similar to the agent learned through the baseline, our adversarial agent also establish weird behaviors and trick the victim into behaving in an undesired fashion. While the abnormal behaviors are similar for these three games, the adversarial agent learned through our method demonstrates stronger exploitability (See Figure 1(a)).

As we also observe from Fig. 1(a), both the baseline and our method fail to identify an effective policy to master the Sumo-Ants game. As is discussed in Gleave et al. (2020), this is because the observation space under the control of an adversary is low, leaving less room for it to exploit the weakness of the victim via its actions. However, we argue, even if we observe the similar winning rate in existing and our approaches, this does not imply our proposed method is as futile as the existing approach. In Figure 1(b), for each game, we show the combination of winning and tie rates (*i.e.,* non-loss rate). As we can observe, for Sumo-Ants, our method could still obtain an adversarial agent that significantly prevents the victim from receiving sufficient wins (*i.e.,* about 88% winning plus tie rates for the adversary).

Following the adversarial agent's exploitability comparison, we also compare the victim agent's behaviors when it confronts two different adversarial agents in this game. As we can observe from the demo videos (`https://tinyurl.com/yxteeyuo`), our adversarial agent



(a) The winning rates of the adversarial agents.



(b) The non-loss (*i.e.,* winning plus tie) rates of the adversarial agents.

Figure 1. The performance comparison of adversarial agents against the victim. The adversarial agents obtained from two different approaches – our proposed method and an existing method (Gleave et al., 2020). Note that the darker solid lines represent the average winning (plus tie) rates, whereas the lighter bands indicate the corresponding variations between the maximal and minimal winning (plus tie) rates. The highlighted y-axis labels are the highest average winning/non-loss rates over the total amount of iterations. In addition to visualizing the winning rate, Supplement Section S6 also conducts a statistical test to further confirm the significance of our method. Note that Fig. S3 in the Supplement further shows the changes in victim agents' winning rate during the attack training.

learns to intentionally jump onto the ground, while the agent learned through the baseline approach keeps still. To understand this behavior difference, we take a closer look at the game rule of the Sumo-Ants game and discover an unfairness design of the game. As we detail in Supplementary Section S5, if any player falls from the arena without contacting its opponent, the game will count the match as a draw. In our experiment, our adversarial agent learns to exploit this rule, preventing the victim agent from winning the game. This result indicates that our attack could find and abuse the game unfairness when the adversary has minimal impact on the victim's observation space. Unlike our attack, the baseline does not demonstrate the capability of exploiting the game's unfairness and thus fail to establish sufficient exploitability against the victim.

Finally, the light color bands in Fig. 1 shows the performance variance of each method. As we can observe that our proposed method generally has less variation in agent performance than the existing technique. It means that our proposed algorithm is more robust against those randomness factors, such as the initial game state and the probabilistic state transition. By following our algorithm, an attacker could, therefore, learn an adversarial agent with consistent performance even if the game initializes his agent at different random places in the adversarial learning process.

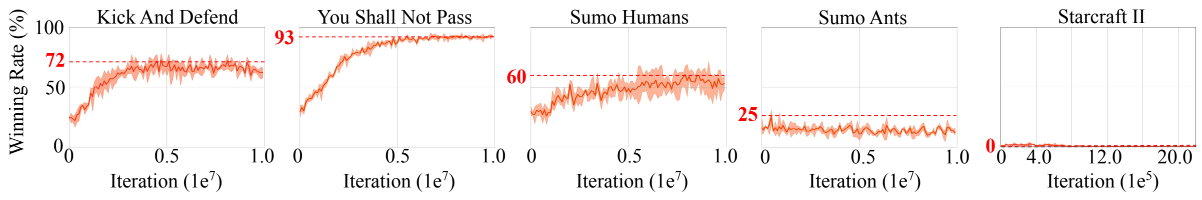
In this work, we also conduct some other comparison experiments. Due to space limit, we present them in the Supplementary Section S6. For example, we compare our method with a method which utilizes only the second term in Eqn. (12) as the objective function (*i.e.,* $\text{argmax}_\theta - \mathbb{E}[\min(\text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t^v, \rho_t A_t^v)]$). This comparison helps assess whether focusing only on reducing the victim's reward can lead to the same attack effect. Besides, we also compare our method with an approach without the monotonic property. This comparison helps us double confirm the importance of monotonicity.

Table 1. The performance of the victims against corresponding regular agents before and after the adversarial retraining. The numbers on the gray canvas represent the winning rates, whereas those on the white one indicate the win plus tie rate.

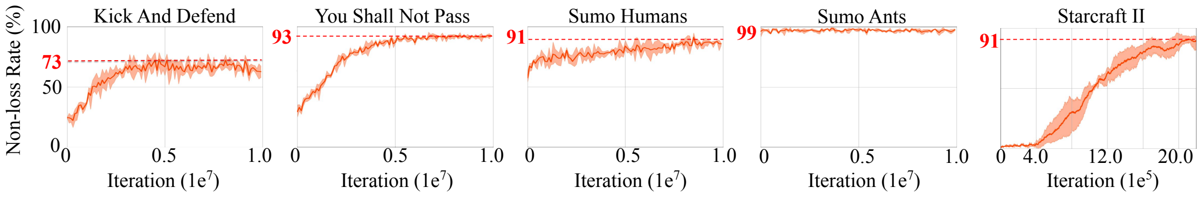
Game	After retraining (%)		Before retraining (%)	
Kick And Defend	12.0	15.0	12.0	14.0
You Shall Not Pass	59.0	59.0	60.0	60.0
Sumo Humans	81.0	92.0	75.0	87.0
Sumo Ants	41.0	59.0	34.0	55.0
StarCraft II	83.0	87.0	46.0	47.0

4.2. Adversary Resistance

Experiment Design. Using the method training the adversary, Gleave et al. (2020) demonstrate that one could retrain the victim and thus improve its adversary resistance. In this experiment, we follow their experimental setup, using the way we train the adversary to retrain the corresponding victim agent. Then, we examine the adversary resistance of the retrained agent. Further, we explore the generalizability of the retrained victim agents. Specifically, we set the retrained victim agent to play with other regular agents for



(a) The winning rates of the adversarial-retrained victim agent against our adversarial agent in five games.



(b) The non-loss rates of the adversarial-retrained victim agent against our adversarial agent in five games.

Figure 2. The performance of victim agents after being retrained by using our proposed learning method.

100 rounds and report its winning rate. Through this setup, we study whether the adversarial training wipes out the retrained agent's ability to play with other regular agents.

Last but not least, we also measure the retrained victim agent's robustness against our attack and its robustness against the baseline attack (Gleave et al., 2020). Specifically, we take the victim agent retrained against our adversarial attack to play with the adversarial agent learned through the baseline and vice versa. In this process, we record the winning rates of the retrained victim agent and examine whether adversarial training against our proposed attack could lead to a more adversary-resistance agent.

Experiment Results. In Figure 2, we show the performance of the victim agent after being retrained.[3] We can observe from the figure that, for all of the five games, the adversarial training takes effect, helping the victim agent pick up the ability to minimize the influence of adversarial agents upon itself. For example, the winning rate of the retrained agent returns 96% in the "You-Shall-Not-Pass" game. For others, the win plus tie rates bounce back to $> 70\%$. As is also shown in table 1, adversarial retraining imposes a negligible influence on the victim agent's ability to play with another regular agent. It indicates that by retraining the victim agent with a mix of adversarial episodes and normal episodes, the retrained agent could establish the adversarial robustness while preserving its generalizability against regular agents. In Supplementary Section S6, we study the influence of the episode split upon the robustness and generalizability of the retrained victim agent.

Table 2 shows the performance of retrained agents against different adversarial agents. As we can first observe from

Table 2. The robustness comparison of the victim retrained against our attack and retrained against the baseline attack (Gleave et al., 2020). The numbers in the table are the win plus tie of the retrained victim agent. "Base" and "adv." refers to "baseline" and "adversary", respectively.

Games	Our retrained victim (%)		Baseline retrained victim (%)	
	vs. Our adv.	vs. Base adv.	vs. Our adv.	vs. Base adv.
Kick And Defend	68.0	66.0	35.0	73.0
You Shall Not Pass	93.0	94.0	86.0	91.0
Sumo Humans	90.0	82.0	71.0	80.0
Sumo Ants	98.0	91.0	91.0	93.0
StarCraft II	89.0	99.0	4.0	87.0

the table that the victim agent retrained against our attack is capable of defeating the adversarial agent obtained by the state-of-art attack (*i.e.,* Column 3). It could achieve the similar robustness to the victim retrained against the baseline (*i.e.,* Column 3 vs. 5). On the contrary, when playing with our adversarial agent, the baseline retrained victim is less effective (*i.e.,* Column 4). It cannot achieve comparable performance with our retrained victim (*i.e.,* Column 1 vs. 4). The result indicates that an attack with a strong exploitability could help the victim agent pick up a robust policy, which is also resistant to an adversarial policy with a weaker exploitability. This finding suggests that users should utilize the more powerful adversarial agent for adversarial retraining and expect the retrained agent could defeat weaker adversaries.

4.3. Root Cause Analysis

Experiment Design. To further understand the root cause behind adversarial policy's effectiveness, we follow Gleave et al. (2020) and conduct the following experiment. First, we blind the victim's observation on the adversary or, in other words, zero out the victim observation pertaining to the adversary. Then, we test the partially blind vic-

[3]Table S1 in Supplement shows the victim agents' performances against our adversarial agents before retraining.

Table 3. The win plus tie rate of the (blind) victim agent against our adversarial agent and that obtained by the baseline attack (Gleave et al., 2020). "Before" and "After" indicates before and after blinding the victim observations, respectively.

Games	Our attack (%)		Baseline attack (%)	
	Before	After	Before	After
Kick And Defend	14.0	94.0	45.0	97.0
You Shall Not Pass	26.0	98.0	48.0	97.0
Sumo Humans	53.0	67.0	65.0	68.0
Sumo Ants	86.0	87.0	91.0	95.0
StarCraft II	2.0	24.0	64.0	98.0

tim against our adversarial agent and that learned from the state-of-the-art method Gleave et al. (2020). Using this experiment, we demonstrate that adversarial agent not only can win by taking actions to induce natural observations that are adversarial to the victim but also can win by exploiting game unfairness.[4]

Experiment Results. Table 3 shows the victim's win plus tie rate before and after blinding. Similar to the findings in Gleave et al. (2020), the victim winning rates against both attacks increase dramatically in the games – You-Shall-Not-Pass and Kick-And-Defend. This result aligns the results present in Gleave et al. (2020). It indicates that, for these two games, the adversarial agent wins the victim by indirectly influencing the victim's observation. However, we also discover the blinding has no impact on our attack in the Sumo-Ants game. This finding confirms that, for this game, our adversarial agent beats the victim by exploiting the game unfairness. Last but not least, our attack establishes a lower adversarial winning rate drop than the existing attack on the Starcraft II games. This confirms that our attack has a stronger exploitability than the existing attack on this sophisticated game.

5. Discussion and Future Work

Transferability. In Supplementary Section S7, we show that both our attack and the state-of-art attack establish a certain level of transferability. Specifically, we find that adversarial policies that explore game unfairness have stronger transferability than those disturbing the victim observation through weird actions. The above observations are obtained when varying only hyperparameters and initial states. Recent works (Huang et al., 2017) on the transferability of the observation manipulation attacks show that the transferability also relies upon many other factors, such as games themselves and learning algorithms. In the future, we will thoroughly evaluate the transferability of our proposed attack under different setups.

Other games. In addition to the two-player Markov game studied in this work, there are two other types of popular competitive games – two-player extensive-form games and two-player multi-agent games (Zhang et al., 2019).[5] Regarding the first type, at any given time step, only one agent can observe the game state and thus take action. As such, the state transition of this type of game depends solely upon the action of one agent. Existing research (Srinivasan et al., 2018) has extended the vanilla policy gradient to train a DRL agent in extensive-form games, such as Poker (Lanctot et al., 2019). Based on the learning method proposed in (Srinivasan et al., 2018), our attack can be potentially generalized to this game. The key challenge is how to enable monotonicity for the adversarial learning algorithm. In the future, we will explore how to design the learning objective function to guarantee monotonicity. Concerning the two-player multi-agent games, the agents in one team cooperate with each other to compete against the agents from the other team. Enabling an adversarial attack in this game is equivalent to training a set of cooperative agents to defeat another set of well-trained agents. As discussed in (Daskalakis et al., 2009), the key challenge of preparing a group of cooperative agents is how to handle the information sharing among the agents. Recent research (Zhang et al., 2018) has explored how to extend the vanilla policy gradient to a multi-agent setting. As part of future work, we will borrow the idea from this technique and generalize our attack to multi-agent settings.

Zero-sum game vs. two-player game. As is mentioned in Section 3.1, in most real-world two-player games, the reward design of the game usually does not follow the rule of zero-sum.[6] As such, our proposed method could demonstrate a performance advantage over the state-of-the-art method (Gleave et al., 2020) (which is built on top of the PPO algorithm). However, we admit that if a two-player game follows the zero-sum rule restrictively, in theory, the adversarial agent trained by our approach will demonstrate the same performance as that prepared by Gleave et al. (2020). In other words, under the zero-sum setup, our objective function could be viewed as maximizing the adversarial expected reward, which is equivalent to the objective function of the PPO method (See Supplement S8 for the demonstration of the equality of our attack and Gleave et al. (2020) in a zero-sum game). While this could potentially become a limitation of our work, we argue this does

[4]In Supplement S6, we also follow Gleave et al. (2020) and conduct an in-depth analysis on the activations gathered from the victim policy networks. The analysis further demonstrates the behavior differences between our adversarial agent and that learned from the attack in Gleave et al. (2020).

[5]Existing research (Daskalakis et al., 2009) has proven that training an RL agent for multi-player games is an NP problem. As such, we do not consider multi-player games in this work.

[6]Such games can also be taken as general-sum competitive games.

not dilute our contribution. As is discussed above, to obtain optimal performance, most two-player games, if not all, do not follow the zero-sum rule strictively. Therefore, it leaves room for an attacker to leverage our approach for exploiting the weakness of agents in the game.

6. Conclusion

In two-player Markov games, existing methods for learning an adversarial policy either make unrealistic assumptions or fail to demonstrate sufficient advantages over target agents (*i.e.,* victim agents), especially when the game does not strictly follow the zero-sum setup. In this work, we develop a new attack against a victim agent trained by DRL in the context of two-player games. Technically, we carefully design the objective function of our adversarial learning algorithm such that the agent trained by our attack could guarantee to increase the expected reward difference between the adversary and victim monotonically. By using five games commonly utilized in reinforcement learning evaluation, we show that our attack not only significantly outperforms the state-of-the-art attack, but also demonstrates an ability to abuse the unfairness of the target game. With this discovery, we safely conclude that our newly proposed attack could help an attacker learn an adversarial agent much more effective in defeating victims.

Acknowledgments

We would like to thank the anonymous reviewers and Meta reviewer for their helpful comments. This project was supported in part by NSF grant CNS-1718459 and CNS-1954466, by ONR grant N00014-20-1-2008, by the Amazon Research Award, and by the IBM Ph.D. Fellowship Award.

References

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

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I. Emergent complexity via multi-agent competition. In *Proc. of ICLR*, 2018.

Behzadan, V. and Munir, A. Vulnerability of deep reinforcement learning to policy induction attacks. In *Proc. of MLDM*, 2017.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *Proc. of S&P*, 2017.

Daskalakis, C., Goldberg, P. W., et al. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 2009.

Gleave, A., Dennis, M., Kant, N., Wild, C., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. In *Proc. of ICLR*, 2020.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *Proc. of ICLR*, 2015.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. In *Proc. of ICLR workshop*, 2017.

Kiourti, P., Wardega, K., Jha, S., and Li, W. Trojdrl: Trojan attacks on deep reinforcement learning agents. *arXiv preprint arXiv:1903.06638*, 2019.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Proc. of NeurIPS*, 2000.

Kos, J. and Song, D. Delving into adversarial attacks on deep policies. In *Proc. of ICLR Workshop*, 2017.

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

Lee, X. Y., Ghadai, S., Tan, K. L., Hegde, C., and Sarkar, S. Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In *Proc. of AAAI*, 2020.

Lin, J., Dzeparoska, K., Zhang, S. Q., Leon-Garcia, A., and Papernot, N. On the robustness of cooperative multi-agent reinforcement learning. In *Proc. of DLS Workshop*, 2020.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. In *Proc. of IJCAI*, 2017.

Lykouris, T., Simchowitz, M., Slivkins, A., and Sun, W. Corruption robust exploration in episodic reinforcement learning. *arXiv preprint arXiv:1911.08689*, 2019.

Ma, Y., Zhang, X., Sun, W., and Zhu, J. Policy poisoning in batch reinforcement learning and control. In *Proc. of NeurIPS*, 2019.

Madry, A., Makelov, A., Schmidt, L., et al. Towards deep learning models resistant to adversarial attacks. In *Proc. of ICLR*, 2018.

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

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *Proc. of EuroS&P*, 2016.

Russo, A. and Proutiere, A. Optimal attacks on reinforcement learning policies. *arXiv preprint arXiv:1907.13548*, 2019.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proc. of ICML*, 2015.

Srinivasan, S., Lanctot, M., Zambaldi, V., Pérolat, J., Tuyls, K., Munos, R., and Bowling, M. Actor-critic policy optimization in partially observable multiagent environments. In *Proc. of NeurIPS*, 2018.

Sun, J., Zhang, T., Xie, X., Ma, L., Zheng, Y., Chen, K., and Liu, Y. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *Proc. of AAAI*, 2020.

Sun, P., Sun, X., Han, L., Xiong, J., Wang, Q., Li, B., Zheng, Y., Liu, J., Liu, Y., Liu, H., et al. Tstarbots: Defeating the cheating level builtin ai in starcraft ii in the full game. *arXiv preprint arXiv:1809.07193*, 2018.

Tesauro, G. Temporal difference learning and td-gammon. *Communications of The ACM*, 1995.

Thrun, S. Monte carlo pomdps. In *Proc. of NeurIPS*, 2000.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Proc. of ICIRS*, 2012.

Unity. Unity machine learning agents toolkit. https://unity3d.com/machine-learning/, 2020.

Xiao, C., Pan, X., He, W., Peng, J., Sun, M., Yi, J., Li, B., and Song, D. Characterizing attacks on deep reinforcement learning. *arXiv preprint arXiv:1907.09470*, 2019.

Yang, Z., Iyer, N., et al. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972*, 2019.

Zhang, H., Chen, H., Boning, D., and Hsieh, C.-J. Robust reinforcement learning on state observations with learned optimal adversary. In *Proc. of ICLR*, 2021.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. Fully decentralized multi-agent reinforcement learning with networked agents. *arXiv preprint arXiv:1802.08757*, 2018.

Zhang, K., Yang, Z., and Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.

Zhao, Y., Shumailov, I., Cui, H., Gao, X., Mullins, R., and Anderson, R. Blackbox attacks on reinforcement learning agents using approximated temporal information. *arXiv preprint arXiv:1909.02918*, 2019.