Dynamics Model Based Adversarial Training For Competitive Reinforcement Learning

Xuan Chen[1], Guanhong Tao[1], Xiangyu Zhang[1]
[1]Purdue University
{chen4124, taog, xyzhang}@cs.purdue.edu

Abstract

Adversarial perturbations substantially degrade the performance of Deep Reinforcement Learning (DRL) agents, reducing the applicability of DRL in practice. Existing adversarial training for robustifying DRL uses the information of agent at the current step to minimize the loss upper bound introduced by adversarial input perturbations. It however only works well for single-agent tasks. The enhanced controversy in two-agent games introduces more dynamics and makes existing methods less effective. Inspired by model-based RL that builds a model for the environment transition probability, we propose a dynamics model based adversarial training framework for modeling multi-step state transitions. Our dynamics model transitively predicts future states, which can provide more precise back-propagated future information during adversarial perturbation generation, and hence improve the agent's empirical robustness substantially under different attacks. Our experiments on four two-agent competitive MuJoCo games show that our method consistently outperforms state-of-the-art adversarial training techniques in terms of empirical robustness and normal functionalities of DRL agents.

1 Introduction

Competitive Reinforcement Learning (CRL) is widely used in autonomous driving [23, 11], automated trading [3] and two(multi)-player games, such as GO [24], Starcraft [27] and Dota 2 [16]. While the power of Deep Neural Networks (DNN) underpins many CRL techniques(e.g., in encoding agents policies), their vulnerability to adversarial attacks [6, 14] in supervised learning tasks emphasizes the need for model robustness in CRL. Existing research shows that a well-trained RL agent can be easily attacked by carefully-designed perturbations in the observation and action spaces [10, 18, 12], or an adversarially-trained opponent [5, 9, 29] that can interact with the victim agent in the environment.

As a remedy, adversarial training has been proposed to improve RL agents robustness. However, most of them focus on single-agent games. [10] studied the robustness of agent with pixel inputs and discrete actions based on [6]. In [31, 15], robustness verification algorithms in image classification tasks were adapted to single-agent games to harden the agent's policy. However in a two-agent competitive RL game, the agents observe each others actions and respond accordingly. Specifically, a player aims to maximize the return while the other aims to minimize it, and vice versa. The existence of an controversial opponent makes existing adversarial training techniques less effective.

In this paper, we propose a novel adversarial training technique for two-agent competitive games. The two players have opposite goals and the sum of the agents' rewards will be zero. It leverages a dynamics model to approximate the state transition function of the underlying environment, which allows predicting the near-future states and generating forward-looking perturbations that can deteriorate victim's long-term performance effectively. Based on the attack, we devise an adversarial training framework that aligns with the intrinsic characteristics of two-player games. Our experiment results show that the proposed algorithm outperforms state-of-the-art RL adversarial training

The First Multi-Agent Security (MASEC) Workshop at NeurIPS 2023

techniques [31, 15] under a wide range of attacks, regarding the robustness and the preservation of performance. Our key contributions include:

- A unique dynamics model based adversarial training framework for two-agent competitive RL games. It enables the victim agent to evaluate future states better and bridges the gap that existing single-agent robust training algorithms cannot achieve satisfactory performance when extended to two-agent scenario.
- Improved empirical robustness demonstrated across 4 different adversarial attacks on 4 different MuJoCo competitive two-agent games, using a new metric to evaluate model robustness in the context of two-player RL games.
- Experiments on attackers with different policies show that our techniques helps the victim generalize the robustness to various adversaries.

Threat Model. We consider the empirical robustness against adversarial perturbations that are added in the observation space, without changing the underlying environment. They could be black-box [31] in which the adversary learns a separate Q-value network to perform attacks, or while-box [10, 13, 31], where the attacker has full knowledge of the victim's policy and value networks. We assume the attacker and the victim follow their original policies during testing, while the attacker can achieve his malicious target by adding perturbations to his own behaviors, which is equivalent to perturbations in the observation space of the the victim agent. The robustness against an opponent with an adversarial policy is out of the scope of this paper.

2 Design

Overview. The overarching idea of our method is to train the victim agent to foresee the future possible adversarial motions of the adversary agent. Intuitively, instead of encouraging the agent to follow its original action, we train the agent to be resilient to *future* adversarial perturbations from the opponent that are able deteriorate the agent's long-term performance. This requires the agent to have a good understanding of "what will happen in the adversarial future" and take action accordingly.

Specifically, we start with generating adversarial perturbations that maximize the victim's long-term performance degradation. A naive method is to maximize the state value of the attacker $V_\alpha(s_\alpha^t)$ while minimizing that of the victim $V_v(s_v^t)$, i.e., maximizing the state value differences at time step t. However, such shortsightedness fails to consider the enhanced controversy between the two parties. We hence take further steps and leverage the state value differences at $t, t+1, ...$ until $t+h$, with h denotes the "*horizon*" of the victim, namely, how much further it can see. To deal with the non-differentiable nature of state transitions, inspired by model-based RL, we learn a dynamics model to mimic the transitions between adjacent states and and predict future states. Then we feed the predicted future states to both agent's value networks to acquire the state value differences of the following time steps. Summing up the state value differences from t to $t+h$ and backpropagating their gradients to the attacker's state at t allows generating forward-looking perturbations at t. As such, we minimize the victim's total state value of immediate-future in the presence of these perturbations. We provide more motivations of our method in Appendix A.

Dynamics Model. The role of a dynamics model in our adversarial training is to help the victim determine future state transitions when both parties take actions according to their policies π_α, π_v. The victim and attacker's state values at $t+1, .., t+h$ are needed to generate perturbation that can maximize the victim's long-period performance degradation while preserving that of the attacker, with h as the hyper-parameter. To predict the future states of both agents, a dynamics model f_ϕ : parameterized through a DNN is trained to predict the environment dynamics. It takes the two agent's states and actions $(s_\alpha^t, s_v^t, a_\alpha^t, a_v^t)$ at time step t as the input, and predicts the difference Δs between the current states and the next time step's states $(s_\alpha^{t+1}, s_v^{t+1})$. The advantage of using the difference as the target output lies in the similarity between two adjacent states, which reduces the difficulty for the neural network to approximate the underlying true dynamics. The training loss of our dynamics model is hence the following.

$$\epsilon(\phi) = \frac{1}{|D|} \sum_{(s_t, a_t, s_{t+1}) \in D} \frac{1}{2} ||(s_{t+1} - s_t) - f_\phi(s_t, a_t)||^2 \tag{1}$$

Here $s_t = (s_\alpha^t, s_v^t), a_t = (a_\alpha^t, a_v^t)$. Intuitively, the model encodes the law of physics, which is implemented by the underlying simulator. The ground-truth training data is collected from trajectory rollouts that are used to train the policy and value networks. We use the states without any adversarial perturbations to avoid noises stemmed from robustness vulnerabilities of the model. Once a dynamics model f_ϕ is learned, we use f_ϕ to transitively estimate the states from $t+1$ until $t+h$.

The dynamics model helps the agent's value network to be more accurate on future state approximation, and thus can help the adversary to generate stronger adversarial perturbations based on better evaluation of near-future state values. Adversarially training the victim agent with the presence of such perturbations hence explicitly extends its horizon, mitigating the shortsightedness problem. See Appendix C for more details about how the dynamics model helps improve future state estimation.

Adversarial Training Framework. As discussed above, the dynamics model transitively predicts $\hat{s}_{t+1}$, ..., $\hat{s}_{t+h}$ when the attacker and the victim follow their current policy. Then we leverage the predicted states to compute the sum of value differences $\sum_{i=t}^{t+h}(V(\hat{s}_\alpha^i) - V(\hat{s}_v^i))$ between the attacker and the victim for the following h time steps. Backpropagating the gradients of the sum wrt. the attacker's current state s_α^t generates forward-looking perturbations δ at t, where $\delta = \epsilon \cdot \text{sign}(\nabla_{s_\alpha^t} \sum_{i=t}^{t+h}(\hat{V}(s_\alpha^i) - \hat{V}(s_v^i))$. Here ϵ controls the l_∞ norm of adversarial perturbation. We add δ to the victim's state input which contains its opponent's information. We also collect the new actions $\hat{a}_v$ induced by perturbations to the trajectory rollouts for later policy training.

The overall adversarial training procedure is similar to the common training process of a clean opponent agent. PPO [22] was implemented to learn the optimal policy for the victim agent. We extend the training objective of PPO by adding adversarial perturbations to the input of victim's policy network, which can be formalized to a minimax optimization problem as follows.

$$\min_\theta \mathbb{E}_{(\hat{s}_v^t, \hat{a}_v^t)\sim\pi_{v_{old}}, r_v^t}[-\min(\frac{\pi_v(\hat{a}_v^t|\hat{s}_v^t)}{\pi_{v_{old}}(\hat{a}_v^t|\hat{s}_v^t)}\hat{A}_v^t, \text{clip}(1-\eta, 1+\eta)\hat{A}_v^t)], \text{ where } \hat{s}_v^t = s_v^t + \Delta s, \tag{2}$$

$$\Delta_s = \text{argmax}_{||\delta||\leq\epsilon}(V_a(s_a^t + \delta) - V_v(s_v^t + \delta)) + \sum_{i=t+1}^{t+h}(V_a(f_\phi(s_a^{(i)})) - V_v(f_\phi(s_v^{(i)}))). \tag{3}$$

Here, $\hat{s}_v^t$ and $\hat{a}_v^t$ denote the perturbed states and actions of the victim agent, respectively, after adding the perturbation δ. The definition of advantage function for policy π is $A_\pi^t(s,a) = Q_\pi(s,a) - V_\pi(s)$, where $Q_\pi(s,a)$ is the Q value for action a in state s [8]. It describes how much better it is to take action a in s over randomly behaving according to π thereafter. We use GAE [21] to estimate the victim's advantage $\hat{A}_v^t$ at t. Note that it is also computed under the perturbed values of the agent. Equation 2 represents the final goal of the victim agent, aiming to maximize its expected total reward (the negative sign transforms it to a minimization problem), while the inner maximization (Equation 3) represents the goal of the adversary to deteriorate its long-term performance.

3 Experiments

3.1 Experimental Setup

We use four competitive two-agent MuJoCo games defined in [1]. We extend four existing adversarial attacks that are applicable on continuous policy to the two-agent tasks and evaluate our proposed method under these attacks. The difference lies in which part of the victim's inputs we are perturbing. In single-agent task, the perturbation are added on the whole input space of the agent, while in two-agent task, we only add noise on the part of the victim's input that denotes the attacker's information, representing the goal of the malicious party to attack by changing its own position. Note that the underlying true states are not changed, the noises are added in the observation space of the victim agent, which is a more realistic setting consistent with the literature [31, 25]. We consider attacks with l_∞ norm as in most literature. See Appendix D for more details about four attacks.

We compare our technique with two state-of-the-art adversarial training baselines, SAPPO [31] and RADIAL [15]. These techniques were originally developed to harden models in single-agent games. We extend them to two-agent games. In particular, we treat the attacker with a fixed stochastic policy as part of the environment of the victim agent, which reduces the two-party game to a single agent game, as how [5] and [9] formalized the problem when training an adversarial agent. Then we apply SAPPO and RADIAL to train the victim agent.

Table 1: Win Rate(%) (absolute difference) of three different adversarial training agents under five different attacks.

Environment	Models	Clean Win Rate	Random	MAD	RS	RS+MAD	VDiff
You shall not pass	PPO	58.7	54.5(-1.2)	23.5(-35.2)	14.1(-44.6)	10.6(-48.1)	28.2(-30.5)
	SAPPO	54.6(-4.1)	56.1(-2.6)	55.2(-3.5)	45.2(-13.5)	**44.6(-14.1)**	56.4(-2.3)
	RADIAL	54.1(-4.6)	55.2(-3.5)	**55.8(-2.9)**	44.8(-13.9)	43.4(-15.3)	56.9(-1.7)
	Our method	**55.8(-2.9)**	**57.8(-0.9)**	55.1(-3.6)	**46.4(-12.3)**	42.9(-15.8)	**57.4(-1.3)**
SumoHuman	PPO	27.9	25.9(-2.0)	10.3(-17.6)	9.4(-18.5)	5.2(-22.7)	12.8(-15.1)
	SAPPO	25.5(-2.4)	25.3(-2.6)	21.0(-6.9)	20.6(-7.3)	21.9(-6.0)	24.8(-3.1)
	RADIAL	24.1(-3.8)	25.4(-2.5)	**28.5(+0.6)**	21.9(-6.0)	20.9(-7.1)	24.8(-3.1)
	Our method	**26.2(-1.7)**	**26.6(-2.4)**	25.8(-2.1)	**22.6(-5.3)**	**22.5(-5.4)**	**26.4(-1.5)**
Kick and Defend	PPO	64.6	59.1(+2.5)	24.5(-40.1)	18.3(-46.3)	15.6(-49.0)	32.3(-32.3)
	SAPPO	55.2(-9.4)	59.4(-5.2)	**50.4(-14.2)**	41.3(-23.3)	40.7(-23.9)	60.5(-4.1)
	RADIAL	56.6(-8.0)	60.4(-4.2)	50.3(-14.3)	40.1(-24.5)	39.7(-24.9)	61.2(-3.4)
	Our method	**57.5(-8.3)**	**62.2(-2.3)**	49.8(-14.8)	**42.3(-22.3)**	**41.1(-23.5)**	**61.4(-3.2)**
SumoAnt	PPO	41.2	38.2(-3.0)	33.1(-8.1)	14.8(-26.4)	10.1(-31.1)	29.5(-11.7)
	SAPPO	34.1(-7.1)	36.8(-4.4)	33.5(-7.7)	24.4(-16.8)	27.3(-13.9)	32.6(-8.6)
	RADIAL	39.1(-2.1)	**42.0(+0.8)**	37.8(-3.4)	26.4(-14.8)	24.2(-17.0)	37.2(-4.0)
	Our method	**39.3(-1.9)**	36.9(-4.3)	**40.8(-0.4)**	**27.5(-13.7)**	**29.5(-11.7)**	**40.5(-0.7)**

Metric. A robust agent should maintain stable and satisfactory performance under any adversarial perturbations. Previous works on single-agent games evaluate the robustness of an agent by the reward value under various attacks. Larger reward values indicate better robustness. However, such metric is insufficient for two-agent games. For example, one may retrain an agent's policy such that it becomes much stronger and achieves a high win rate (i.e., beating its opponent). The resulted larger reward values do not mean that its policy becomes more robust.

We propose to use the absolute win rate difference, i.e., the win rate when the agent is playing with a benign opponent and the win rate playing with an opponent under adversarial attack, as the metric to evaluate robustness. Intuitively, it illustrates how the policy is resilient to adversarial perturbations, regardless the strength of the policy itself. In addition to the win rate difference, we also look at whether our adversarial training incurs win rate degradation (when compared to without adversarial training). We obtain the win rate of the victim agent over 500 random seeds.

3.2 Experimental Results

In this section, we show that our methods produces victim agents with better robustness and less performance degradation than two state-of-the-art adversarial training algorithms on four MuJoCo competitive two-agent games.

The results are shown in Table 1. The first column describes the games. The second column presents the models with PPO denoting the victim policy without any adversarial training, SAPPO, RADIAL, and our method denoting models hardened by respective methods. The third column presents the clean win rate (without any adversarial attack). The remaining columns denoting the results for various attacks, including Random in which the attacker sampled the noise from a uniform distribution, MAD, RS, RS+MAD, and VDiff. The numbers in braces denote the dimension of the observation of each game. Observe that RS+MAD can achieve almost strongest attacks on victim's observation space, our adversarial training can largely improve the agent's empirical robustness against a wide range of attacks and maintain high clean win rate on three games comparing with two baselines. We also perform the ablation study on our key hyper-parameter h in Appendix D.

4 Conclusion

In this work we study the empirical robustness problem against adversarial attacks that are applied in the context of competitive two-agent games. We propose a dynamics model based adversarial training method that can improve the victim agent's robustness against four existing adversarial attacks. A potential future direction is to use dynamics model that incorporates environment uncertainty to achieve better performance.

References

[1] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018.

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

[3] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[4] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[5] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[7] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[8] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

[9] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. Adversarial policy learning in two-player competitive games. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

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

[11] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[12] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[13] 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 *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3756–3762, 2017.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[15] Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel, and Tsui-Wei Weng. Robust deep reinforcement learning through adversarial loss. In *Advances in Neural Information Processing Systems*, 2021.

[16] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[17] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

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

[19] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[20] Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning, 2022.

[21] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

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

[23] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[24] 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*, 529(7587):484–489, 2016.

[25] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep RL. In *International Conference on Learning Representations*, 2022.

[26] Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[27] 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*, 575(7782):350–354, 2019.

[28] Tsui-Wei Weng, Krishnamurthy (Dj) Dvijotham*, Jonathan Uesato*, Kai Xiao*, Sven Gowal*, Robert Stanforth*, and Pushmeet Kohli. Toward evaluating robustness of deep reinforcement learning with continuous control. In *International Conference on Learning Representations*, 2020.

[29] Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1883–1900, 2021.

[30] Huan Zhang, Hongge Chen, Duane S. Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021.

[31] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Advances in Neural Information Processing Systems*, 2020.

[32] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

A Motivation

In this section, we use an example to illustrate the limitations of existing adversarial training for deep RL agents and motivate our work.

A Two-agent Competitive Game and Robustness Our work focuses on two-agent competitive games. The two players in such games have opposite goals and they compete with each other. They are also called zero-sum games since the sum of the agents' rewards will be zero. Both parties have their own independent policy network, whose input is the observation of that agent (regarding the other agent and the environment) and the output is the action the agent will take. In Figure 1, we use the simulated robotics game *You Shall Not Pass* defined in [1] as an example. There are two players: the runner (in red) and the blocker (in blue). The runner wins if it reaches the finish line; the blocker wins otherwise. An agent can observe its opponent's information, for example, the locations of different joints, and use it as the input to the policy network.

We aim to improve the robustness of policy networks used in two-agent games. Specifically, we study *how an agent remains robust if its opponent is adding adversarial perturbations to its behaviors (e.g., changing joint positions).* Environmental perturbations are hence beyond our threat model. Ideally, a robust victim agent would be able to maintain its original performance, when it faces the malicious agent, and be resilient to the perturbations that the adversary adds to itself.

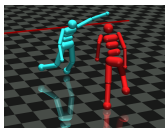

Figure 1: You Shall Not Pass

Limitations of Existing RL Model Hardening Methods. Most existing works (e.g., [31, 15]) focus on training a robust RL agent in single-agent games. SAPPO [31] first generates the upper bound of current action deviation when there are input perturbations *at the current time step* and then updates the model weights to minimize that upper bound when the perturbations are applied. RADIAL-PPO[15] incorporates the upper bound of the perturbed loss to the final training loss of the victim agent. Minimizing such loss practically discourages the overlap between good action and bad action *at the current time step* under adversarial perturbation.

These techniques derive perturbations based on the input and the state of the current step. This may not be sufficient for two-agent model hardening as two-agent games are much more dynamic due to their zero-sum nature. In Figure 2, for illustrative purposes, we show the top view of both agents to display the movement trajectory. We treat the blue runner as the attacker and the red blocker as the victim. The attacker uses PGD [14] to generate adversarial perturbations and adds them to its joints. The victim has been adversarially trained using SAPPO. Under the attack, the victim agent tends to aggressively move towards the attacker (illustrated by the yellow arrows in the figure), even though a better strategy may be to move side-way to block the attacker's trajectory. It suggests that the victim focuses on the situation at the current moment and lacks the ability to evaluate the future possible results that the attacker exploits the aggressive move of the blocker and eventually bypasses it. It is understandable since during adversarial training, the victim's goal is to follow its original action (in the presence of perturbation), which was to move towards the attacker. However, the original action was out-dated due to the perturbations. In single-agent games, such near-sighted weakness does not pose as severe threat as it would in two-agent games, since there is not a proactively adjusting opponent. It is hence sufficient to stay in course given environmental perturbations.

Figure 3 presents the same initial states as Figure 2. The difference is that the victim has been adversarially trained by our method. Observe that instead of running towards the attacker aggressively, the victim moves side way to block the attacker's trajectory. In other words, it has better anticipation of the attacker's movement, enabled by the dynamics model.

B Backgrounds

In this section, we define the notations used in describing our adversarial training technique. We use (α, v) to represent the attacker and the victim agents, respectively. A two-player competitive game can be represented as a Markov Decision Process (MDP) $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S} = (\mathcal{S}_\alpha, \mathcal{S}_v), \mathcal{A} = (\mathcal{A}_\alpha, \mathcal{A}_v)$ denote the state sets and the action sets of the attacker and the victim, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ denotes the probability of transition between two states. Given the state $s = (s^t_\alpha, s^t_v)$ and action $a = (a^t_\alpha, a^t_v)$ at time step t, the transition probability can be written as $p(s' \mid s, a) =$

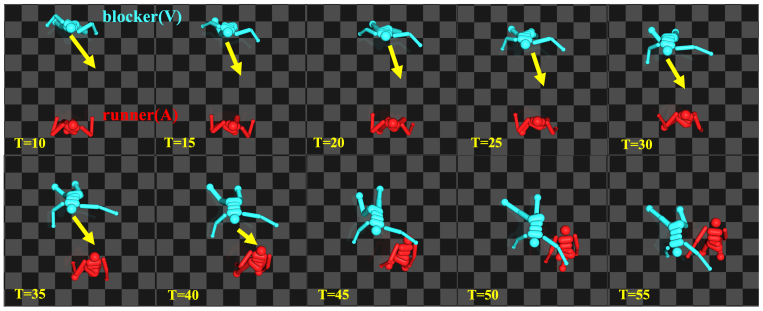

Figure 2: Top view of the *You Shall Not Pass* game. The victim blocker (in blue) has been adversarially trained by SAPPO and the runner (in red) is malicious, with its joint positions perturbed by PGD to maximize the chance of penetrating the blocker. The yellow arrow denotes the moving direction of the victim's center of mass, i.e., the heading direction of its body. The attacker successfully passes the blocker.

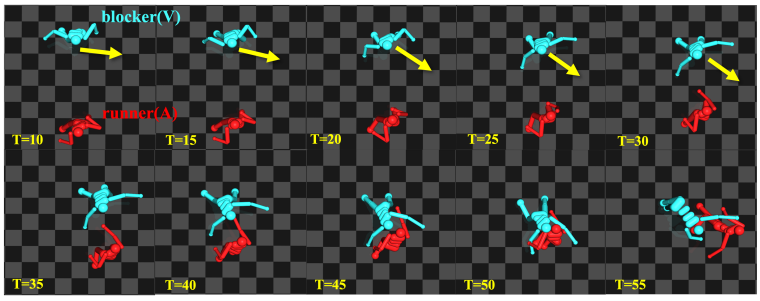

Figure 3: The same initial game states as Figure 2, with the victim blocker (in blue) adversarially trained by our method. The blocker successfully stops the adversarial runner with perturbations generated by PGD.

$\Pr\{S_{t+1} = s' | S_t = s, A_t = a\}$. P is also said defining the *dynamics* of MDP. The reward function $R = (R_\alpha, R_v)$, with $R_i : \mathcal{S}_i \times \mathcal{A}_i \to \mathbb{R}$ ($i \in \{\alpha, v\}$) denoting a scalar function that outputs the numerical reward that an agent receives when it transits from s to s' with action a.

Parameter $\gamma \in [0, 1]$ is a discounting factor. In the framework of MDP, the goal of training an agent in a competitive game is to learn a policy $\pi(a|s) : (\mathcal{S} \times \mathcal{A} \to \mathbb{R})$, which is a mapping from states to probabilities of selecting each possible action. This is by maximizing the expected total return $\mathbb{E}_{a\sim\pi(a|s)}[\sum_t^\infty \gamma^t R(s, a)]$. when the agent behaves according to π. Mathematically, the above objective can be translated to maximizing the state value function of policy π defined as follows.

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s], \text{for all } s \in \mathcal{S} \tag{4}$$

The value function of a state s under a policy π, is the expected total return when the agent starts in s and follows π thereafter, representing how good it is for the agent to be in the current state. Although constructing the perturbation based on the victim's value function $V_\pi(s_t)$ at time step t reflects the intention of deteriorating the agent's long-term performance, looking forward solely based on the value function at t is insufficient as we will show in the experiment.

In our work, we assume the victim uses a fixed policy π_θ approximated by a neural network with parameters θ during test time.

C Additional Technical Details

Dynamic model. Figure 4 shows the mean squared error between the ground truth state value $V(s_{t+1})$ and predicted $V(\hat{s}_{t+1})$ generated with and without dynamics model when the blocker agent is at time step t. This error describes the value estimation quality of future state $\hat{s}_{t+1}$. Specifically, the ground-truth state values $V(s_{t+1})$ are collected from the trajectory. In the first case, we use the dynamics model to predict $\hat{s}_{t+1}$ and feed it to the value network to get $V(\hat{s}_{t+1})$. In the second case (i.e., without the dynamics model), we acquire $V(\hat{s}_{t+1})$ using one-step *Temporal Difference* (TD) learning [26] that we normally do when training the value network. We predict the future states recursively using the following equation:

$$s_\alpha^{t+h}, s_v^{t+h} = \begin{cases} s_\alpha^t, s_v^t & h = 0 \\ (s_\alpha^{t+h-1}, s_v^{t+h-1}) + f_\phi(s_\alpha^{t+h-1}, s_v^{t+h-1}, a_\alpha^{t+h-1}, a_v^{t+h-1}) & h > 0 \end{cases} \tag{5}$$

Improving Future State Estimation by the Dynamics Model. The dynamics model helps the agent's value network to be more accurate on future state approximation, and thus can help the adversary to generate stronger adversarial perturbations based on better evaluation of near-future state values. Adversarially training the victim agent with the presence of such perturbations hence explicitly extends its horizon, mitigating the shortsightedness problem. Figure 4 shows the mean squared error between the ground truth state value $V(s_{t+1})$ and predicted $\hat{V}(s_{t+1})$ generated with and without dynamics model when the blocker agent is at time step t. This error describes the value estimation quality of future state $\hat{s}_{t+1}$. Specifically, the ground-truth state values $V(s_{t+1})$ are collected from the trajectory. In the first case, we use the dynamics model to predict $\hat{s}_{t+1}$ and feed it to the value network to get $V(\hat{s}_{t+1})$. In the second case (i.e., without the dynamics model), we acquire $V(\hat{s}_{t+1})$ using one-step *Temporal Difference* (TD) learning [26] that we normally do when training the value network. Experiment details are included in the Appendix. In Figure 4, we compute the above value errors for the first 50 time steps over 50 trajectories with different random seeds. The shadow area denotes the standard deviations. Existing research [4] on model-based RL has proven that incorporating an ideal dynamics model that can achieve h-depth accuracy to model-free value estimation algorithms and leveraging these synthetic states to update the value network improve it to be accurate on imagined states compared with the original value network, which explains why our adversarial perturbation can help reach better robustness to some extend.

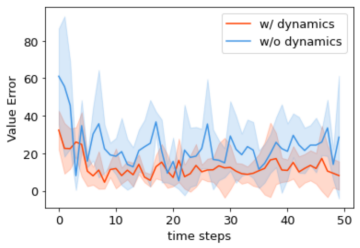

Figure 4: Comparison of value estimation error on imagined state $\hat{s}_{t+1}$.

D Additional Experiment Details And Results

For the VDiff attack in [18], we generate perturbations to minimize the state value of the victim agent at t. Maximal Action Difference (MAD) in [31] maximizes the deviation between the original action and the perturbed action by maximizing the KL-divergence between the original and perturbed policies. Robust Sarsa (RS) [31] learns a robust Q network to lead the agent to take the worst action that has the lowest Q value. RS+MAD [31] combines the above two techniques together, using a weight of loss to control the balance between the two attacks. We use the perturbation bound of 1.5 and the attack steps of 5. The l_∞ used in our experiment is larger than the commonly-used bound 0.15 in existing adversarial training methods for single-agent games [31, 25], since we are only perturbing part of the victim's state observation, requiring more aggressive perturbation to achieve the attack goal. Note that we use the same bound in all experiments for fair comparison.

Ablation Study. The horizon h in dynamics model controls how much further we would like the perturbation to deteriorate the victim's performance. Intuitively, larger h leads to perturbations with longer effect on the victim's performance but the estimation quality for far-future states will be sacrificed. We conduct ablation study on the influence of the hyper-parameter horizon used in the dynamics model on the game *SumoHuman*. Based on Tabel 2, we could see that $h = 1$ achieves the best result for SumoHuman agents, while $h = 5$ degrades the victim's performance and the reason could be that larger horizon will induce inaccurate prediction of future states. The robustness without dynamics model, i.e. $h = 0$, is worse than $h = 1$, when we look further for one step during adversarial training.

Table 2: Ablation Study on Horizon h, table shows the win rate of *SumoHumans* agent that is adversarially trained by dynamics models with different horizon h under 5 adversarial attacks.

Adversarial Attacks	$h = 1$	$h = 2$	$h = 3$	$h = 5$
Random	21.2	26.2	23.8	18.6
MAD	20.9	25.8	23.2	18.4
RS	19.6	23.1	20.3	17.8
RS+MAD	18.3	21.7	19.8	17.6
VDiff	20.0	26.4	23.0	17.5

E Related Work

E.1 RL Model Hardening Methods

For single-agent tasks, [12, 2, 18] proposed to improve agent's robustness using perturbations that are based on adversarial example attacks against deep learning models [6, 17] in supervised learning, since the discrete action outputs can be viewed as labels. [19] focused on the robustness against environment parameter settings(mass, friction values), considering the attacker as another agent that can apply forces on the victim with continuous actions. [31, 15] studied the robustness against full observation space perturbation leveraging the neural network verification techniques [32, 7]. [30, 25] consider robustness against the optimal adversarial perturbations and train an agent whose state space depends on the victim agent to stress the challenge.

For CRL tasks, [5] showed that attacker can induce out-of-distribution observation of the victim, which can be used to retrain the victim policy. [9] further improved the effectiveness of adversarial agents, considering imposing negative influence on victim agent besides improving the reward of the attacker. Although they focused on two-agent games, we are under different threat models and have different goals and evaluation methods.

E.2 Adversarial attacks on RL agents

A line of works [10, 13, 28, 12] proposed to manipulate victim's whole observation to maximize the action change under perturbation or to decrease the quality of action that the victim is going to take. [28] also incorporate a learned dynamics model to perform adversarial attacks. In contrast, our method directly utilizes the gradients based on the differentiable nature of the dynamics model to generate perturbations for adversarial training.

F Discussion

While our current DNN-parameterized dynamics model is effective, exploring other architectures or hybrid models may yield better predictive accuracy, especially in complex environments. Also, for some environments with high dimension observation and action space, it is difficult to precisely predict the future states, and the prediction errors will accumulated with the increase of h. In the future, we will explore how to incorporate the more advanced model-based RL techniques [20] to further improve the performance of the dynamics model and extend to more practical scenarios.