On the Robustness of Safe Reinforcement Learning under Observational Perturbations

Zuxin Liu[1], **Zijian Guo**[1], **Zhepeng Cen**[1], **Huan Zhang**[1], **Jie Tan**[2], **Bo Li**[3], **Ding Zhao**[1]
[1]CMU, [2] Google Brain, [3] UIUC
{zuxinl, zijiang, zcen, huanzhan}@andrew.cmu.edu
jietan@google.edu, lbo@illinois.edu, dingzhao@cmu.edu

Abstract

Safe reinforcement learning (RL) trains a policy to maximize the task reward while satisfying safety constraints. While prior works focus on performance optimality, we find that the optimal solutions of many safe RL problems are not robust and safe against observational perturbations. We formally analyze the unique properties of designing effective state adversarial attackers in the safe RL setting. We show that baseline adversarial attack techniques for standard RL tasks are not always effective for safe RL and proposed two new approaches - one maximizes the cost and the other maximizes the reward. One interesting and counter-intuitive finding is that the maximum reward attack is strong, as it can both induce unsafe behaviors and make the attack stealthy by maintaining the reward. We further propose a more effective adversarial training framework for safe RL and evaluate it via comprehensive experiments [1]. This paper provides a pioneer work to investigate the safety and robustness of RL under observational attacks for future safe RL studies.

1 Introduction

Despite the success of deep reinforcement learning (RL), it is still challenging to ensure safety when deploying them to real-world applications. Safe RL tackles the problem by solving a constrained optimization that can maximize the task reward while satisfying certain constraints [1–5]. The success of recent safe RL approaches leverages the power of neural networks (NNs) [6, 7]. However, it has been shown that NNs are vulnerable to adversarial attacks [8, 9], which raises a concern when deploying an NN-based RL policy to safety-critical scenarios [10]. We consider the observational perturbations that commonly exist in the physical world, such as unavoidable sensor errors and perception inaccuracy [11]. Several recent works of observational robust RL propose attackers that can drastically decrease their rewards [12, 13]. However, the methods in standard RL settings may not be suitable for safe RL because of an additional metric that characterizes the cost of constraint violations [14]. The cost should be more important than the measure of reward since any constraint violations could be fatal and unacceptable in the real world [15]. We find little research formally studying the robustness in the safe RL setting with adversarial observation perturbations, while we believe this should be an important aspect in the safe RL area, because **a vulnerable policy under adversarial attacks cannot be regarded as truly safe in the physical world.**

We aim to address the following questions in this work: 1) How vulnerable would a learned RL agent be under observational adversarial attacks? 2) How to design effective attackers in the safe RL setting? 3) How to obtain a robust policy that can maintain safety even under worst-case perturbations? To answer them, we formally define the observational robust safe RL problem and discuss how to evaluate the adversary and robustness of a safe RL policy. We also propose two strong adversarial attacks that can induce the agent to perform unsafe behaviors and show that adversarial training can help improve the robustness of constraint satisfaction. We summarize the contributions as follows.

[1]video demos are available at: https://sites.google.com/view/robustsaferl/home

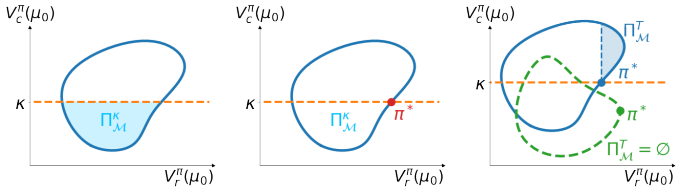
ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

1. As far as we are aware, we are the first to formally analyze the unique vulnerability of the optimal policy in safe RL under observational corruptions. We define the state-adversarial safe RL problem and investigate its fundamental properties. We show that the optimal solutions of safe RL problems are theoretically vulnerable under observational adversarial attacks.
2. We show that existing adversarial attack algorithms focusing on minimizing agent rewards do not always work, and propose two effective attack algorithms. Surprisingly, the maximum reward attack is very strong in inducing unsafe behaviors, both in theory and practice.
3. We propose an adversarial training algorithm for safe RL. Extensive experiments show that our method is more robust against adversarial perturbations regarding constraint satisfaction.

2 State Adversarial Safe RL

2.1 CMDP and the safe RL problem

A Constrained Markov Decision Process (CMDP) [16] is defined by the tuple $(S; A; P; r; c; \ ; \ _0)$, where S is the state space, A is the action space, P is the transition probability, r is the reward function, c is the cost function for violating constraints, $\ $ is the discount factor, and $\ _0$ is the initial state distribution. We denote a safe RL problem as $M_\Pi := (S; A; P; r; c; \ ; \ _0; \ ; \ )$, where $\ $ is the policy class, and $\ $ is a threshold for constraint violation cost. Let $(ajs) \ 2 \ $ denote the policy and $\ = fs_0; a_0; :::; g$ denote the trajectory. We use shorthand $f_t = f(s_t; a_t; s_{t+1}); f \ 2 \ fr; cg$ for simplicity. The value function is $V_f(\ _0) = E_{\ ;s_0 \ \ _0}[\sum_{t=0}^{1} \ ^t f_t]; f \ 2 \ fr; cg$, which is the expectation of discounted return under the policy $\ $ and the initial state distribution $\ _0$. We denote $Q_f(s; a) = E_{\ ;s_0=s;a_0=a}[\sum_{t=0}^{1} \ ^t f_t]; f \ 2 \ fr; cg$ as the state-action value function under the policy $\ $. The objective of M_Π is to find the policy that maximizes the reward while limiting the cost incurred from constraint violations to a threshold $\ $: $\ = \arg\max \ V_r(\ _0); \ s.t. \ V_c(\ _0) \ $. We denote the **feasible** policy class as the set of policies that satisfies the constraint with threshold $\ $: $\ _M := f(ajs) : V_c(\ _0) \ \ ; \ 2 \ g$; the **optimal** policy $\ $ as the policy with the highest reward return in the feasible policy class: $\ 2 \ _M; 8 \ \ 2 \ \ _M; V_r(\ _0) \ V_r(\ _0)$; the **tempting** policy class as the set of policies that have a higher reward return than the optimal policy: $\ _M^T := f(ajs) : V_r(\ _0) > V_r(\ _0); \ 2 \ g$. The figure illustration is shown in Fig. 1. Note that although the temptation concept naturally exists in many safe RL settings, we did not find formal descriptions or definitions of it in the literature. We show that all the tempting policies are unsafe:



(a) Feasibility (b) Optimality (c) Temptation

Figure 1: Illustration of definitions via a mapping from the policy space to the metric plane $\Pi \to R^2$, where the x-axis is the reward return and the y-axis is the cost return. A point on the metric plane denotes corresponding policies with the reward and cost return.

Lemma 1. *The tempting policy class and the feasible policy class are disjoint:* $\ _M^T \ \setminus \ _M = ;$. *Namely, all the tempting policies violate the constraint:* $8 \ \ 2 \ \ _M^T; V_c(\ _0) > \ $.

See proof in Appendix A.1. The existence of tempting policies is a unique feature and one challenge of safe RL since the agent needs to maximize the reward carefully to avoid being tempted.

2.2 Safe RL under observational perturbations

We introduce a deterministic **observational** adversary $\ (s) : S \ ! \ S$ which corrupts the state observation of the agent. We denote the corrupted state as $\tilde{s} := \ (s)$ and the corrupted policy as $\tilde{} := \ (ajs) = \ (aj \ (s))$, as the state is first contaminated by $\ $ and then used by the operator $\ $. Furthermore, we restrict the power of the adversary by defining the perturbation set $B_p(s)$ as the $`_p$-ball around the original state: $8s' \ 2 \ B_p(s); ks' \ \ sk_p \ \ \ $, where $\ $ is the ball size.

A strong attacker in safe RL should be 1) **effective** in increasing the constraint violation cost and 2) **stealthy** such that the perturbations can hardly be identified. While the stealthiness regarding the state perturbation range is naturally satisfied based on the perturbation set definition, we introduce another level of stealthiness in terms of the task reward in the safe RL task. In some situations, the agent might easily detect a dramatic reward drop, so a more stealthy attack maintains the agent's task reward while increasing constraint violations; see Appendix B.1 for more discussions.

Motivated by Lemma 1, we propose the following **Maximum Reward (MR) attacker** that corrupts the observation of a policy π by maximizing the reward value: $\nu_{\text{MR}} = \arg\max_{\nu} V_r(\pi, \nu(s_0))$

Proposition 1. *For an optimal policy π^*, the MR attacker is guaranteed to be reward stealthy and effective, given enough large perturbation set $B_p(s)$ such that $V_r(\pi, \nu_{\text{MR}}) > V_r(\pi^*)$.*

The MR attacker is counter-intuitive because it is exactly the goal for standard RL.This is an interesting phenomenon worthy of highlighting since we observe that the MR attacker effectively makes the optimal policy unsafe and retains stealthy regarding the reward in the safe RL setting. The proof is in Appendix A.1. We further observe the following important property for the optimal policy:

Lemma 2. *The optimal policy π^* of a tempting safe RL problem satisfies: $V_c(\pi^*, \nu(s_0)) = d$.*

The proof is given in Appendix A.2. It suggests that the optimal policy in a tempting safe RL problem is vulnerable as it is on the safety boundary, which motivates us to propose the **Maximum Cost (MC) attacker** that corrupts the observation of a policy π by maximizing the cost value: $\nu_{\text{MC}} = \arg\max_{\nu} V_c(\pi, \nu(s_0))$. It is apparent to see that the MC attacker is effective since we directly solve the adversarial state such that it can maximize the constraint violations. The two proposed attacker implementation details can be found in Appendix C.1. The theoretical analysis of adversarial attacks such as the cost value bound can be found in Appendix A.3.

2.3 Observational robust safe RL

To defend against observational perturbations, we propose an adversarial training method for safe RL, which directly optimize the policy upon the attacked sampling trajectories $\tau \sim \{s_0, a_0, s_1, a_1, \ldots\}$, where $a_t \sim \pi(a|\nu(s_t))$. We can compactly represent the adversarial safe RL objective under observational perturbation as: $\pi_{\text{adv}} = \arg\max_{\pi} V_r(\pi, \nu(s_0)), \; s.t. \; V_c(\pi, \nu(s_0)) \leq d$. It can be solved by many policy-based safe RL methods [2, 3], and we show that the Bellman operator for evaluating the policy performance under a deterministic adversary is a contraction (see Appendix A.7 for proof).

Theorem 1 (Bellman contraction)**.** *Define the Bellman policy operator as $T_{\nu}: \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$:*

$$(T_{\nu} V_f)(s) = \sum_{a \in A} \pi(a|\nu(s)) \sum_{s' \in S} p(s'|s,a) \left[f(s,a,s') + \gamma V_f(s') \right], \quad f \in \{r, c\}. \tag{1}$$

The Bellman equation can be written as $V_f(s) = (T_{\nu} V_f)(s)$. In addition, the operator T_{ν} is a contraction under the sup-norm $\|\cdot\|_1$ and has a fixed point.

Theorem 1 provides the theoretical justification of adversarial training, since we can accurately evaluate the reward and cost values of a policy under one fixed deterministic adversary. Then the key part is selecting proper adversaries during learning, such that the trained policy is robust and safe against any other attackers. We show that performing adversarial training with the MC or the MR attacker will enable the agent to be robust against the most effective or the most reward stealthy perturbations, respectively (see Appendix A.7 for detailed discussions and proof). The adversarial training algorithm and implementation details could be found in Appendix C.3.

3 Experiment

We aim to answer the questions raised in Sec. 1. To this end, we adopt the constrained robot locomotion continuous control tasks that are used in many previous works [17–20]: `Car-Run`, `Car-Circle`, `Drone-Run`, `Drone-Circle`, `Ant-Run`, `Ant-Circle`. The detailed description is in Appendix C.6. We use the PID PPO-Lagrangian (abbreviated as PPOL) method [3] as the base safe RL algorithm to fairly compare different robust training approaches, while the proposed methods can be easily used in other safe RL methods as well (see Appendix C.8 for more results with other safe RL methods: FOCOPS [19], SAC-Lagrangian [21], and CVPO [5]). The detailed hyperparameters of the adversaries and safe RL algorithms can be found in Appendix C.

Adversarial attacker baselines. We adopt three adversary baselines: 1) **Random attacker**, which samples the corrupted observations randomly within the perturbation set; 2) **Maximum Action Difference (MAD) attacker** [11] for standard RL tasks, which is shown to be effective in decreasing a trained RL agent's reward return; 3) **Adaptive MAD (AMAD) attacker**, which is an improved version of MAD by performing attacks only in high-risk regions (motivated by Lemma 2. See Appendix C.4 for derivation and implementation details of the attacker baselines.

Robust training baselines. We adopt 5 baselines, including **PPOL-vanilla** without robust training, the training under random noise **PPOL-random**, the state-adversarial algorithm **SA-PPOL** proposed in [11]. The original SA-PPOL utilizes the MAD attacker to perform adversarial training, while

we add two additional baselines **SA-PPOL(MC)** and **SA-PPOL(MR)** for the ablation study, where the attacker is replaced by our proposed MC and MR attackers. Our adversarial training methods are named as **ADV-PPOL(MC)** and **ADV-PPOL(MR)**, which are trained under the MC and MR attackers respectively. More details can be found in Appendix C.5-C.7.

Attackers comparison. Fig. 2 shows the attack results of the 5 adversaries on PPOL-vanilla. The first row is the reward and the second row is the cost of constraint violations. We can see that the vanilla safe RL policies are vulnerable, since the safety performance deteriorates (cost increases) significantly. Although baselines can reduce the reward, they fail to perform an effective attack in increasing the cost. Our proposed MC and MR attackers outperform all baselines by increasing the cost by a large margin in most tasks. Surprisingly, MR can achieve even higher costs than MC and is more stealthy as it can maintain or increase the reward.

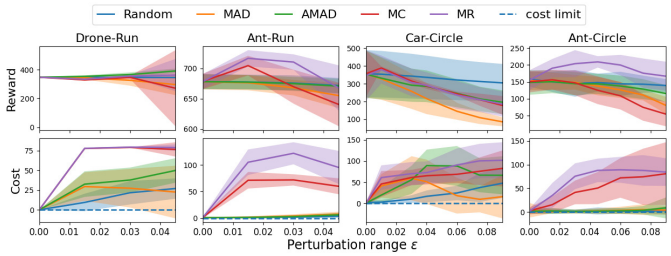


Figure 2: Reward and cost curves of all 5 attackers evaluated on well-trained vanilla PPO-Lagrangian models w.r.t. the perturbation range .

Table 1: Evaluation results of natural performance (no attack) and under 3 attackers. Our methods are ADV-PPOL(MC/MR). Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds. We shadow two lowest-costs agents under each attacker column and break ties based on rewards, excluding the failing agents (whose natural rewards are less than 50% of PPOL-vanilla's). We mark the failing agents with *?*.

Env	Method	Natural		AMAD		MC		MR	
		Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Ant-Run = 0:025	PPOL-vanilla	678.4 12.64	1.23 1.4	676.23 12.27	2.68 2.16	661.3 58.17	66.41 14.07	706.32 18.83	112.33 25.57
	PPOL-random	673.42 14.47	1.01 1.06	670.6 13.59	1.9 1.47	661.47 10.02	45.94 10.2	670.85 18.73	46.97 11.63
	SA-PPOL	658.83 14.14	0.46 0.82	658.42 13.96	0.66 0.87	668.14 25.7	67.68 20.17	694.86 11.05	87.1 20.78
	SA-PPOL(MC)	574.36 25.69	3.03 3.15	574.85 26.37	3.16 3.48	604.77 30.51	21.39 10.83	619.4 31.35	32.87 12.69
	? SA-PPOL(MR)	90.49 60.14	5.33 4.27	77.64 84.93	5.17 4.24	77.99 72.79	6.33 4.87	69.93 96.51	6.17 4.87
	ADV-PPOL(MC)	601.25 18.6	0.0 0.0	599.31 18.34	0.0 0.0	666.73 15.21	1.1 1.02	665.47 18.29	1.75 1.54
	ADV-PPOL(MR)	620.17 27.28	0.17 0.41	618.04 24.66	0.31 0.55	634.96 14.94	4.07 2.35	648.95 17.67	4.69 2.81
Ant Circle = 0:025	PPOL-vanilla	157.44 26.21	2.7 6.02	143.37 36.86	3.23 9.87	153.98 34.52	38.93 29.78	208.81 20.1	70.53 22.6
	PPOL-random	155.81 16.84	2.67 6.6	150.65 17.63	2.17 5.02	114.24 35.22	1.83 6.08	183.07 24.63	58.53 22.3
	SA-PPOL	143.34 32.08	0.13 0.56	142.66 35.01	4.53 10.67	159.02 43.95	37.47 26.5	203.85 27.56	51.47 27.79
	? SA-PPOL(MC)	-0.62 1.72	0.0 0.0	0.09 1.27	0.0 0.0	-0.17 1.46	0.0 0.0	-0.34 1.61	0.0 0.0
	? SA-PPOL(MR)	-0.8 2.28	0.0 0.0	-0.57 2.2	0.0 0.0	-0.89 2.12	0.0 0.0	-0.86 2.09	0.0 0.0
	ADV-PPOL(MC)	135.98 15.99	0.3 1.62	130.76 18.87	0.77 4.13	137.13 29.4	6.33 13.96	134.68 22.01	5.3 9.39
	ADV-PPOL(MR)	133.27 19.53	0.87 3.25	127.19 32.64	1.2 4.49	118.57 26.37	0.83 2.02	141.74 23.63	1.07 3.08

Robust training comparison. The results of different trained policies under adversarial attacks are shown in Table 1, where **Natural** is the noise-free performance. Due to page limits, we leave the results in the other four environments and the results under random and MAD attackers to Appendix C.8. We can observe that although most baselines can achieve near zero natural cost, their safety performances are vulnerable to the proposed MC and MR attackers. The proposed adversarial training methods (ADV-PPOL) consistently outperform baselines in safety with the lowest costs while maintaining high rewards in most tasks. The comparison with PPOL-random indicates that the MC and MR attackers are essential ingredients of adversarial training. Although SA-PPOL agents can maintain reward very well, they are not safe as to constraint satisfaction under adversarial perturbations in most environments. The ablation studies with SA-PPOL(MC) and SA-PPOL(MR) suggest that the KL-regularized robust training technique, which is successful in standard robust RL setting, does not work well for safe RL even with the same adversarial attacks during training, and they may also fail to obtain a high-rewarding policy in some tasks (see discussions of the training failure in Appendix B.2). As a result, we can conclude that the proposed adversarial training methods with the MC and MR attackers are better than baselines regarding both training stability and testing robustness and safety.

Conclusion We study the observational robustness regarding constraint satisfaction for safe RL and show that the optimal policy of tempting safe RL problems could be vulnerable. We propose two effective attackers to induce unsafe behaviors. An interesting and surprising finding is that maximizing the reward attack is as effective as directly maximizing the cost while keeping stealthiness. We further propose an adversarial training method to increase the robustness and safety performance for safe RL, and extensive experiments show that the proposed method outperforms the robust training techniques for standard RL settings. Our results show the existence of a previously unrecognized problem in safe RL, and we hope this work encourages other researchers to study safety from the robustness perspective, as both safety and robustness are important ingredients for real-world deployment.

References

[1] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5, 2021.

[2] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019.

[3] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[4] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

[5] Zuxin Liu, Zhepeng Cen, Vladislav Isenbaev, Wei Liu, Steven Wu, Bo Li, and Ding Zhao. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pages 13644–13668. PMLR, 2022.

[6] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

[7] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

[8] Gabriel Resende Machado, Eugênio Silva, and Ronaldo Ribeiro Goldschmidt. Adversarial machine learning in image classification: A survey toward the defender's perspective. *ACM Computing Surveys (CSUR)*, 55(1):1–38, 2021.

[9] Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34:100199, 2019.

[10] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[11] 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. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

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

[13] 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 H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21024–21037. Curran Associates, Inc., 2020.

[14] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[15] Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *arXiv preprint arXiv:1705.08551*, 2017.

[16] Eitan Altman. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical methods of operations research*, 48(3):387–417, 1998.

[17] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[18] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[19] Yiming Zhang, Quan Vuong, and Keith Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 2020.

[20] Sven Gronauer. Bullet-safety-gym: Aframework for constrained reinforcement learning. 2022.

[21] Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. In *AAAI*, pages 10639–10646, 2021.

[22] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

[23] Amram Meir and Emmett Keeler. A theorem on contraction mappings. *Journal of Mathematical Analysis and Applications*, 28(2):326–329, 1969.

[24] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

[27] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[28] Santiago Paternain, Luiz FO Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. *arXiv preprint arXiv:1910.13393*, 2019.

[29] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[30] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[31] 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.

[32] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in Neural Information Processing Systems*, 33:13049–13061, 2020.

[33] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

[34] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G Bellemare. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016.

[35] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Checklist

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

A X-Risk Sheet

Individual question responses do not decisively imply relevance or irrelevance to existential risk reduction. Do not check a box if it is not applicable.

A.1 Long-Term Impact on Advanced AI Systems

In this section, please analyze how this work shapes the process that will lead to advanced AI systems and how it steers the process in a safer direction.

1. **Overview.** How is this work intended to reduce existential risks from advanced AI systems?
   Answer: Though deep RL has witnessed great success in solving challenging problems, the potential risks arise while deploying deep RL in real-world applications, among which **safety** and **robustness** are two non-negligible factors. This work studies RL safety by enforcing the learning agent to satisfy certain constraints, such that the safety performance could be evaluated separately from the task performance (reward). Though the recent advances in safe RL effectively solve challenging RL problems with safety constraints, can we guarantee that the learned policy is really safe in the real world? The answer is probably no if we only consider the constraints during policy optimization, because in this work, we find that **the learned safe RL policies are vulnerable to adversarial attacks both in theory and practice**: a small perturbation in the observation space could induce dangerous behaviors of the agent and cause a dramatic drop in the safety performance. Therefore, a well-trained safe RL policy in the noise-free simulation environment may not be truly safe for real-world deployment due to the commonly existing sensing noises, and thus improving its robustness against observational perturbations is of equal importance as learning a constraint satisfaction policy.

   We propose two effective attack algorithms with theoretical justifications for constrained RL – one directly maximizes the constraint violation cost, and one maximizes the task reward to induce a tempting but risky policy. We surprisingly find that the maximum reward attack is very strong in inducing unsafe behaviors. **We believe this property is overlooked as maximizing reward is the optimization goal for standard RL, yet it leads to risky and stealthy attacks to safety constraints**. Since this attack can maintain the nominal reward, it may not be detected in practice before catastrophic failures. For example, in some real-world applications, task-related metrics (such as velocity, acceleration, goal distances) are usually easy to be monitored from sensors. However, the safety metrics can be sparse and hard to monitor until breaking the constraints, such as colliding with obstacles and entering hazard states, which are determined by binary indicator signals. Therefore, a dramatic task-related metrics (reward) drop might be easily detected by the agent, while constraint violation signals could be hard to detect until catastrophic failures. Our experiment results also show that all the vanilla safe RL methods suffer the vulnerability issue – though they are safe in noise-free environments, they are not safe anymore under strong attacks. The finding of the stealthy and effective maximum reward attack brings up the potential risks of deploying safe RL methods in safety-critical applications, and again validates the necessity of studying the observational robustness of safe RL agents.

   Based on the above findings regarding attacks, we further propose an adversarial training mechanism based on the attacks to reduce the risks for safe RL deployment. The proposed robust training can be served as a plugin for any on-policy safe RL algorithms in principle, which is simple yet effective. Our results show that the robustified policies can achieve much safer performance under strong attacks while maintaining good task performance. We believe this work sheds light on the inherited connection between observational robustness and safety in RL and provides a pioneer work for future safe RL studies.

2. **Direct Effects.** If this work directly reduces existential risks, what are the main hazards, vulnerabilities, or failure modes that it directly affects?
   Answer: Constraint violations of AI agents, the vulnerability of safety performance under noises, AI inducing dangerous behaviors and catastrophic failures.

3. **Diffuse Effects.** If this work reduces existential risks indirectly or diffusely, what are the main contributing factors that it affects?
   Answer: Simulation to the real-world transfer of learning agents, generalization capability of AI agents to unseen domains.

4. **What's at Stake?** What is a future scenario in which this research direction could prevent the sudden, large-scale loss of life? If not applicable, what is a future scenario in which this research direction be highly beneficial?
   Answer: This research direction can benefit a large-scale of deep RL agents that will be deployed in real life, such as self-driving vehicles and household robots. For example, consider the autonomous vehicle navigation task where the reward is to reach the goal as fast as possible and the safety constraint is to not collide with obstacles, then sacrificing some reward is not comparable with violating the constraint because the latter may cause catastrophic consequences. However, our results suggest that under carefully-designed adversarial attacks to the sensing system, the vehicle may falsely decide to violate traffic rules or even collide with pedestrians to maximize the reward. Unfortunately, we find little research formally studying the robustness in the safe RL setting with adversarial observation perturbations, while we believe this should be an important aspect in the safe RL area because **a vulnerable policy under adversarial attacks cannot be regarded as truly safe in the physical world.** Our work can hopefully inspire more research in this direction by considering both the safety and the robustness of the learning AI agents.
5. **Result Fragility.** Do the findings rest on strong theoretical assumptions; are they not demonstrated using leading-edge tasks or models; or are the findings highly sensitive to hyperparameters?
6. **Problem Difficulty.** Is it implausible that any practical system could ever markedly outperform humans at this task?
7. **Human Unreliability.** Does this approach strongly depend on handcrafted features, expert supervision, or human reliability?
8. **Competitive Pressures.** Does work towards this approach strongly trade off against raw intelligence, other general capabilities, or economic utility?

A.2 Safety-Capabilities Balance

In this section, please analyze how this work relates to general capabilities and how it affects the balance between safety and hazards from general capabilities.

9. **Overview.** How does this improve safety more than it improves general capabilities?
   Answer: This work suggests the natural balance between safety performance (defined by cost) and task performance (defined by reward) by introducing the tempting policy class definition. Recall that the tempting policies have higher reward returns than the optimal policy but are unsafe. Therefore, the goal of safe RL is to find a policy that can maximize the reward without being tempted. In addition, we separate task rewards and constraint violation costs in the problem formulation. Combining the cost with reward to a single scalar metric, which can be viewed as manually selecting Lagrange multipliers, may work in simple problems. However, it lacks interpretability – it is hard to explain what a single scalar value means, and requires good domain knowledge of the problem – the weight between costs and rewards should be carefully balanced, which is difficult when the task rewards already contain many objectives/factors. On the other hand, separating the costs from rewards is easy to monitor the safety performance and task performance, respectively, which is more interpretable and applicable for different cost constraint thresholds.
10. **Red Teaming.** What is a way in which this hastens general capabilities or the onset of x-risks?
    Answer: This work finds that improving the robustness and safety performance under adversarial attacks may induce a certain loss of task performance. In other words, the robustified agent may have conservative behavior. However, we believe a small range of reward loss is acceptable when compared to the large margin of safety performance improvement. We also believe that with a larger model capacity and carefully designed safe RL optimization techniques, this type of reward loss could be mitigated further.
11. **General Tasks.** Does this work advance progress on tasks that have been previously considered the subject of usual capabilities research?
12. **General Goals.** Does this improve or facilitate research towards general prediction, classification, state estimation, efficiency, scalability, generation, data compression, executing clear instructions, helpfulness, informativeness, reasoning, planning, researching, optimization, (self-)supervised learning, sequential decision making, recursive self-improvement, open-ended goals, models accessing the Internet, or similar capabilities?

13. **Correlation With General Aptitude.** Is the analyzed capability known to be highly predicted by general cognitive ability or educational attainment?
14. **Safety via Capabilities.** Does this advance safety along with, or as a consequence of, advancing other capabilities or the study of AI?

A.3 Elaborations and Other Considerations

15. **Other.** What clarifications or uncertainties about this work and x-risk are worth mentioning?
    Answer: This trustworthiness of an RL agent doesn't only contain robustness and safety, and other aspects such as generalization and interpretability are also important as well. We believe that the conceptions between them have non-negligible overlaps, though they are mostly discussed as orthogonal directions in the RL literature. For example, we could view robustness regarding observation noises as a generalization capability to the adversarial states around the training samples. Similarly, a generalizable policy for uncertain environments can be regarded as the robustness property to unseen scenarios. Therefore, those concepts are inseparable in a certain context, and we can also see similar discussions and thoughts in the other machine learning domain other than RL. In addition, safety is closely related to the other aspects, because an agent cannot be regarded as safe if it is not generalizable to novel situations or robust against sensing noises. Our work unveils the connections between robustness and safety, and we hope to see more interdisciplinary discussions in this direction.

Contents

A Proofs and Discussions

A.1 Proof of Lemma 1 and Proposition 1 – infeasible tempting policies

Lemma 1 indicates that all the tempting policies are infeasible: $\forall \pi \in \Pi_M^T, V_c(\pi_0) > \epsilon$. We will prove it by contradiction.

Proof. For a tempting safe RL problem M_Π, there exists a tempting policy that satisfies the constraint: $\pi' \in \Pi_M^T, V_c^{\pi'}(\pi_0) \leq \epsilon, \pi' \in \Pi_M$. Denote the optimal policy as π^*, then based on the definition of the tempting policy, we have $V_r^{\pi'}(\pi_0) > V_r^{\pi^*}(\pi_0)$. Based on the definition of optimality, we know that for any other feasible policy $\pi \in \Pi_M$, we have:

$$V_r^{\pi'}(\pi_0) > V_r^{\pi^*}(\pi_0) \geq V_r^{\pi}(\pi_0),$$

which indicates that π' is the optimal policy for M_Π. Then again, based on the definition of tempting policy, we will obtain:

$$V_r^{\pi'}(\pi_0) > V_r^{\pi'}(\pi_0),$$

which contradicts to the fact that $V_r^{\pi'}(\pi_0) = V_r^{\pi'}(\pi_0)$. Therefore, there is no tempting policy that satisfies the constraint. □

Proposition 1 suggest that as long as the MR attacker can successfully obtain a policy that has higher reward return than the optimal policy given enough large perturbation set $B_p(s)$, it is guaranteed to be reward stealthy and effective.

Proof. The stealthiness is naturally satisfied based on the definition. The effectiveness is guaranteed by Lemma 1. Since the corrupted policy π^{MR} can achieve $V_r^{\pi^{\text{MR}}} > V_r^{\pi^*}$, we can conclude that π^{MR} is within the tempting policy class, since it has higher reward than the optimal policy. Then we know that it will violate the constraint based on Lemma 1, and thus the MR attacker is effective. □

A.2 Proof of Lemma 2 – optimal policy's cost value

Lemma 2 says that the optimal policy of a tempting safe RL problem satisfies: $V_c(\pi_0) = \epsilon$. We will also prove it by contradiction.

Proof. Suppose the optimal policy for a tempting safe RL problem M_Π has: $V_c(\pi_0) < \epsilon$. Denote $\pi' \in \Pi_M^T$ as a tempting policy. Based on Lemma 1, we know that $V_c^{\pi'}(\pi_0) > \epsilon$ and $V_r^{\pi'}(\pi_0) > V_r^{\pi^*}(\pi_0)$. Then we can compute a weight ω:

$$\omega = \frac{V_c(\pi_0) - \epsilon}{V_c^{\pi'}(\pi_0) - V_c(\pi_0)}. \tag{2}$$

We can see that:

$$\omega V_c^{\pi'}(\pi_0) + (1-\omega) V_c(\pi_0) = \epsilon. \tag{3}$$

We further define another policy π'' based on the mixture of π^* and π', such that a trajectory of a whole episode has ω probability to be sampled from π' and $1-\omega$ probability to be sampled from π^*:

$$\pi'' := \begin{cases} \pi'; & \text{with probability } \omega; \\ \pi^*; & \text{with probability } 1-\omega. \end{cases} \tag{4}$$

Then we can conclude that π'' is also feasible:

$$V_c^{\pi''}(\pi_0) = \mathbb{E}_{\pi''}\left[\sum_{t=0}^{1} \gamma^t c_t\right] = \omega \mathbb{E}_{\pi'}\left[\sum_{t=0}^{1} \gamma^t c_t\right] + (1-\omega)\mathbb{E}_{\pi^*}\left[\sum_{t=0}^{1} \gamma^t c_t\right] \tag{5}$$

$$= \omega V_c^{\pi'}(\pi_0) + (1-\omega) V_c(\pi_0) = \epsilon. \tag{6}$$

In addition, has higher reward return than the optimal policy :

$$V_r^{-}(\ _0) = \mathsf{E}\ _{-}[\sum_{t=0}^{1}\ ^t r_t] = \ \mathsf{E}\ _{\ 0}[\sum_{t=0}^{1}\ ^t r_t] + (1\ \ )\mathsf{E}\ \ [\sum_{t=0}^{1}\ ^t r_t] \tag{7}$$

$$= \ V_r\ ^{0}(\ _0) + (1\ \ )V_r\ \ (\ _0) \tag{8}$$

$$> \ V_r\ \ (\ _0) + (1\ \ )V_r\ \ (\ _0) = V_r\ \ (\ _0); \tag{9}$$

where the inequality comes from the definition of the tempting policy. Since is both feasible, and has strictly higher reward return than the policy , we know that is not optimal, which contradicts to our assumption. Therefore, the optimal policy should always satisfy $V_c\ \ (\ _0) = \ $.

□

Remark 1. The cost value function $V_c\ \ (\ _0) = \mathsf{E}\ \ [\sum_{t=0}^{1}\ ^t c_t]$ is based on the expectation of the sampled trajectories (expectation over episodes) rather than a single trajectory (expectation within one episode), because for a single sampled trajectory , $V_c\ \ (\ ) = \sum_{t=0}^{1}\ ^t c_t$ may even not necessarily satisfy the constraint.

Remark 2. The proof also indicates that the range of metric function $V := f(V_r\ (\ _0); V_c\ (\ _0))g$ (as shown as the blue circle in Fig.1) is convex when enlarging the domain from to a linear policy mixture space . For simplicity, we define $h\alpha; \pi i$ as a policy mixture 2 which samples mixed trajectories episodically,

$$h\alpha; \pi i := \ \ _i; \text{ with probability } \ _i; i = 1; 2; : : : ; \tag{10}$$

where $\alpha = [\ _1; \ _2; : : :]; \ _i \ \ 0; \sum_{i=1}\ \ _i = 1; \pi = [\ _1; \ _2; : : :]$. Similar to the above proof, we have

$$V_f^{h\alpha; \pi i}(\ _0) = h\alpha; V_f^{\pi}(\ _0)i; f \ 2 \ fr; cg; \tag{11}$$

where $V_f^{\pi}(\ _0) = [V_f\ ^1(\ _0); V_f\ ^2(\ _0); : : :]$. Consider $8(v_{r1}; v_{c1}); (v_{r2}; v_{c2}) \ 2 \ V$, suppose they correspond to policy mixture $h\alpha; \pi i$ and $h\beta; \pi i$ respectively, then $8t \ 2 \ [0; 1]$, the new mixture $ht\alpha + (1 \ \ t)\beta; \pi i \ 2$ and $V_f^{ht\alpha+(1\ \ t)\beta; \pi i}(\ _0) = t \ \ v_{f1} + (1 \ \ t) \ \ v_{f2} \ 2 \ V$. Therefore, V is a convex set.

A.3 Theoretical analysis of adversarial attacks

We have the following theorem to guarantee the existence of optimal and deterministic MC and MR adversaries.

Theorem 2 (Existence of optimal and deterministic MC/MR attackers)**.** *A deterministic MC attacker* $_{MC}$ *and a deterministic MR attacker* $_{MR}$ *always exist, and there is no stochastic adversary* ^{0} *such that* $V_c\ \ ^{0}(\ _0) > V_c\ \ ^{MC}(\ _0)$ *or* $V_r\ \ ^{0}(\ _0) > V_r\ \ ^{MR}(\ _0)$.

Theorem 2 provides the theoretical foundation of Bellman operators that require optimal and deterministic adversaries in the next section. The proof is given in Appendix A.4. We can also obtain the upper-bound of constraint violations of the adversary attack at state s. Denote S_c as the set of unsafe states that have non-zero cost: $S_c := fs^{0} \ 2 \ S : c(s; a; s^{0}) > 0g$ and p_s as the maximum probability of entering unsafe states from state s: $p_s = \max_a \sum_{s^{0}2S_c} p(s^{0}js; a)$.

Theorem 3 (One-step perturbation cost value bound)**.** *Suppose the optimal policy is locally* L*-Lipschitz continuous at state* s*:* $D_{\mathrm{TV}}[\ (js^{0})k\ (js)] \ \ L \ ks^{0} \ \ sk_p$*, and the perturbation set of the adversary* (s) *is an* $`_p$*-ball* $B_p(s)$*. Let* $\nabla_{c\ ;}\ (s) = \mathsf{E}_{a\ \ (j\ (s)); s^{0}\ \ p(\ js; a)}[c(s; a; s^{0}) + \ V_c\ (s^{0})]$ *denote the cost value for only perturbing state* s*. The upper bound of* $\nabla_{c\ ;}\ (s)$ *is given by:*

$$\nabla_{c\ ;}\ (s) \ \ V_c\ (s) \ \ 2L\ \left(p_s C_m + \frac{C_m}{1}\right): \tag{12}$$

Note that $\nabla_{c\ ;}\ (s) \ \ V_c\ (\ (s))$ because the next state s^{0} is still transited from the original state s, i.e., $s^{0} \ \ p(\ js; a)$ instead of $s^{0} \ \ p(\ j\ (s); a)$. Theorem 3 indicates that the power of an adversary is controlled by the policy smoothness L and perturbation range . In addition, the p_s term indicates that a safe policy should keep a safe distance from the unsafe state to prevent it from being attacked. We further derive the upper bound of constraint violation for attacking the entire episodes.

Theorem 4 (Episodic perturbation cost value bound)**.** *Given a feasible policy $\in \Pi_M$, suppose the L-Lipschitz continuity holds globally for , and the perturbation set of is within an ℓ_p-ball, then the following bound holds:*

$$V_c \quad (\ _0) \qquad + 2L\ C_m \left(\frac{1}{1} + \frac{4\ L}{(1 \quad)^2}\right)\left(\max_s p_s + \frac{}{1}\right): \tag{13}$$

See Theorem 3, 4 proofs in Appendix A.5, A.6. We can still observe that the maximum cost value under perturbations is bounded by the Lipschitzness of the policy and the maximum perturbation range . The bound is tight since when $\to 0$ (no attack) or $L \to 0$ (constant policy $(\cdot|s)$ for all states), the RHS is 0 for Eq. (12) and for Eq. (13), which means that the attack is ineffective.

A.4 Proof of Theorem 2 – existence of optimal deterministic MC/MR adversary

Existence. Given a fixed policy , We first introduce two adversary MDPs $\hat{M}_r = (S, \hat{A}, \hat{P}, \hat{R}_r,)$ for reward maximization adversary and $\hat{M}_c = (S, \hat{A}, \hat{P}, \hat{R}_c,)$ for cost maximization adversary to prove the existence of optimal adversary. In adversary MDPs, the adversary acts as the agent to choose a perturbed state as the action (i.e., $\hat{a} = s$) to maximize the cumulative reward $\sum \hat{R}$. Therefore, in adversary MDPs, the action space $\hat{A} = S$ and $(\cdot|s)$ denotes a policy distribution.

Based on the above definitions, we can also derive transition function and reward function for new MDPs [11]

$$\hat{p}(s'|s,\hat{a}) = \sum_a (a|s,\hat{a})p(s'|s,a), \tag{14}$$

$$\hat{R}_f(s,\hat{a},s') = \begin{cases} \frac{\sum_a (a|s,\hat{a})p(s'|s,a)f(s,a,s')}{\sum_a (a|s,\hat{a})p(s'|s,a)}, & \hat{a} \in B_p(s) \\ C, & \hat{a} \notin B_p(s) \end{cases}; f \in \{r, c\}, \tag{15}$$

where $\hat{a} = s$ $(\cdot|s)$ and C is a constant. Therefore, with sufficiently large C, we can guarantee that the optimal adversary will not choose a perturbed state $\hat{a}$ out of the l_p-ball of the given state s, i.e., $(\hat{a}|s) = 0, \forall \hat{a} \notin B_p(s)$.

According to the properties of MDP [22], $\hat{M}_r, \hat{M}_c$ have corresponding optimal policy $_r, \ _c$, which are deterministic by assigning unit mass probability to the optimal action $\hat{a}$ for each state.

Next, we will prove that $_r = \ _{\text{MR}}, \ _c = \ _{\text{MC}}$. Consider value function in $\hat{M}_f, f \in \{r, c\}$, for an adversary $\in \mathcal{N} := \{ \mid (\hat{a}|s) = 0, \forall \hat{a} \notin B_p(s)\}$, we have

$$\hat{V}_f\ (s) = \mathbb{E}_{\hat{a} \quad (\cdot|s), s' \quad \hat{p}(\cdot|s,\hat{a})}[\hat{R}_f(s,\hat{a},s') + \quad \hat{V}_f\ (s')] \tag{16}$$

$$= \sum_{\hat{a}} (\hat{a}|s) \sum_{s'} \hat{p}(s'|s,\hat{a})[\hat{R}_f(s,\hat{a},s') + \quad \hat{V}_f\ (s')] \tag{17}$$

$$= \sum_{\hat{a}} (\hat{a}|s) \sum_{s'} \sum_a (a|s,\hat{a})p(s'|s,a)\left[\frac{\sum_a (a|s,\hat{a})p(s'|s,a)f(s,a,s')}{\sum_a (a|s,\hat{a})p(s'|s,a)} + \quad \hat{V}_f\ (s')\right] \tag{18}$$

$$= \sum_{s'} p(s'|s,a) \sum_a (a|s,\hat{a}) \sum_{\hat{a}} (\hat{a}|s)[f(s,a,s') + \quad \hat{V}_f\ (s')] \tag{19}$$

$$= \sum_{s'} p(s'|s,a) \sum_a (a| \ (s))[f(s,a,s') + \quad \hat{V}_f\ (s')]. \tag{20}$$

Recall the value function in original safe RL problem,

$$V_f \quad (s) = \sum_{s'} p(s'|s,a) \sum_a (a| \ (s))[f(s,a,s') + \quad V_f \quad (s')]. \tag{21}$$

Therefore, $V_f \quad (s) = \hat{V}_f\ (s), \quad \in \mathcal{N}$. Note that in adversary MDPs $_f \in \mathcal{N}$ and

$$_f = \arg\max \mathbb{E}_{a \quad (\cdot| \ (s)), s' \quad p(\cdot|s,a)}[f(s,a,s') + \quad \hat{V}_f\ (s')]. \tag{22}$$

We also know that $_f$ is deterministic,

$$) \quad _f(s) = \arg\max \mathsf{E}_a \quad _{(\,j\bar{s});s'} \quad p(\,js;a)[f(s;a;s') + \quad \hat{V}_f \ (s')] \tag{23}$$

$$= \arg\max \mathsf{E}_a \quad _{(\,j\bar{s});s'} \quad p(\,js;a)[f(s;a;s') + \quad V_f \quad (s')] \tag{24}$$

$$= \arg\max V_f \quad (s;a): \tag{25}$$

Therefore, $_r = \ _{\mathrm{MR}}; \ _c = \ _{\mathrm{MC}}$.

Optimality. We will prove the optimality by contradiction. By definition, $8s \ 2 \ S;$

$$V_c \quad ^{\prime}(s_0) \quad V_c \quad ^{\mathrm{MC}}(s_0): \tag{26}$$

Suppose $9 \ ^{\prime}$, $s{:}t{:}V_c \quad ^{\prime}(\ _0) > V_c \quad ^{\mathrm{MC}}(\ _0)$, then there also exists $s_0 \ 2 \ S$, $s{:}t{:}V_c \quad ^{\prime}(s_0) > V_c \quad ^{\mathrm{MC}}(s_0)$, which is contradictory to Eq.(26). Similarly, we can also prove that the property holds for $_{\mathrm{MR}}$ by replacing V_c with V_r. Therefore, there is no other adversary that achieves higher attack effectiveness than $_{\mathrm{MR}}$ or higher reward stealthiness than $_{\mathrm{MR}}$.

A.5 Proof of Theorem 3 – one-step attack cost bound

We have

$$V_c \ ; \ (s) = \mathsf{E}_a \quad _{(\,j \ (s));s'} \quad p(\,js;a)[c(s;a;s') + \quad V_c \ (s')]: \tag{27}$$

By Bellman equation,

$$V_c \ (s) = \mathsf{E}_a \quad _{(\,js);s'} \quad p(\,js;a)[c(s;a;s') + \quad V \ (s')]: \tag{28}$$

For simplicity, denote $p_{sa}^{s'} = p(s'js;a)$ and we have

$$\nabla_c \ ; \ (s) \quad \nabla_c \ (s) = \sum_{a2A} \left(\quad (aj \ (s)) \quad \quad (ajs) \sum_{s2S} p_{sa}^{s'}(c(s;a;s') + \quad V_c \ (s')) \right) \tag{29}$$

$$\left(\sum_{a2A} j \ (aj \ (s)) \quad \quad (ajs)j \right) \max_{a2A} \sum_{s2S} p_{sa}^{s'}(c(s;a;s') + \quad V_c \ (s')): \tag{30}$$

By definition, $D_{\mathrm{TV}}[\ (j \ (s)k \ (\,js)] = \frac{1}{2} \sum_{a2A} j \ (aj \ (s)) \quad \quad (ajs)j$, and $c(s;a;s') = 0; s' \ 2 \ S_c$. Therefore, we have

$$\nabla_c \ ; \ (s) \quad V_c \ (s) \quad 2D_{TV}[\ (j \ (s)k \ (\,js)] \max_{a2A} \left(\sum_{s2S_c} p_{sa}^{s'} c(s;a;s') + \sum_{s2S} p_{sa}^{s'} \ V_c \ (s') \right) \tag{31}$$

$$2Lk \ (s) \quad sk_p \max_{a2A} \left(\sum_{s2S_c} p_{sa}^{s'} C_m + \sum_{s2S} p_{sa}^{s'} \ \frac{C_m}{1} \right) \tag{32}$$

$$2L \ \left(p_s C_m + \frac{C_m}{1} \right): \tag{33}$$

A.6 Proof of Theorem 4 – episodic attack cost bound

According to the Corollary 2 in CPO [17],

$$V_c \quad (\ _0) \quad V_c \ (\ _0) \quad \frac{1}{1} \mathsf{E}_{s \ d \ ;a} \quad \left[A_c(s;a) + \frac{2 \ _c}{1} D_{\mathrm{TV}}[\ ^{\prime}(\,js)k \ (\,js)] \right]; \tag{34}$$

where $_c = \max_s jE_a \quad A_c(s;a)j$ and $A_c(s;a) = \mathsf{E}_{s'} \ _{p(\,js;a)}[c(s;a;s') + \quad V_c \ (s') \quad V_c \ (s)]$ denotes the advantage function. Note that

$$\mathsf{E}_a \quad A_c(s;a) = \mathsf{E}_a \quad [\mathsf{E}_{s'} \ _{p(\,js;a)}[c(s;a;s') + \quad V_c \ (s') \quad V_c \ (s)]] \tag{35}$$

$$= \mathsf{E}_a \quad _{;s'} \ _{p(\,js;a)}[c(s;a;s') + \quad V_c \ (s')] \quad V_c \ (s) \tag{36}$$

$$= \nabla_c \ ; \ (s) \quad V_c \ (s): \tag{37}$$

By theorem 3,

$$_c = \max_s jE_a \quad A_c(s;a)j \tag{38}$$

$$\max_s \left| 2L \quad \left(p_s C_m + \frac{C_m}{1} \right) \right| \tag{39}$$

$$= 2L \; C_m \left(\max_s p_s + \frac{}{1} \right) : \tag{40}$$

Therefore, we can derive

$$V_c \quad (\ _0) \quad V_c (\ _0) \quad \frac{1}{1} \max_s jE_a \quad A_c(s;a)j + \frac{2 \ _c}{(1 \quad)^2} D_{\mathrm{TV}}[\ ^0(js)k \ (js)] \tag{41}$$

$$= \left(\frac{1}{1} + \frac{2 \ D_{\mathrm{TV}}}{(1 \quad)^2} \right) \ _c \tag{42}$$

$$2L \; C_m \left(\frac{1}{1} + \frac{4 \ L}{(1 \quad)^2} \right) \left(\max_s p_s + \frac{}{1} \right) : \tag{43}$$

Note is a feasible policy, i.e., $V_c (\ _0)$. Therefore,

$$V_c \quad (\ _0) \quad + 2L \; C_m \left(\frac{1}{1} + \frac{4 \ L}{(1 \quad)^2} \right) \left(\max_s p_s + \frac{}{1} \right) : \tag{44}$$

A.7 Proof of Theorem 1 and Proposition 2 – Bellman contraction

Recall Theorem 1, the Bellman policy operator T is a contraction under the sup-norm $k \ k_1$ and will converge to its fixed point. The Bellman policy operator is defined as:

$$(T \ V_f \quad)(s) = \sum_{a2A} \quad (aj \ (s)) \sum_{s^0 2S} p(s^0 js;a) \left[f(s;a;s^0) + \quad V_f \quad (s^0) \right] ; \quad f \ 2 \ fr;cg; \tag{45}$$

The proof is as follows:

Proof. Denote $f_{sa}^{s^0} = f(s;a;s^0); f \ 2 \ fr;cg$ and $p_{sa}^{s^0} = p(s^0 js;a)$ for simplicity, we have:

$$\left| (T \ U_f \quad)(s) \quad (T \ V_f \quad)(s) \right| = \left| \sum_{a2A} \quad (aj \ (s)) \sum_{s^0 2S} p_{sa}^{s^0} \left[f_{sa}^{s^0} + \quad U_f \quad (s^0) \right] \right. \tag{46}$$

$$\left. \sum_{a2A} \quad (aj \ (s)) \sum_{s^0 2S} p_{sa}^{s^0} \left[f_{sa}^{s^0} + \quad V_f \quad (s^0) \right] \right| \tag{47}$$

$$= \left| \sum_{a2A} \quad (aj \ (s)) \sum_{s^0 2S} p_{sa}^{s^0} \left[U_f \quad (s^0) \quad V_f \quad (s^0) \right] \right| \tag{48}$$

$$\max_{s^0 2S} \left| U_f \quad (s^0) \quad V_f \quad (s^0) \right| \tag{49}$$

$$= \left\| U_f \quad (s^0) \quad V_f \quad (s^0) \right\|_1 ; \tag{50}$$

Since the above holds for any state s, we have:

$$\max_s \left| (T \ U_f \quad)(s) \quad (T \ V_f \quad)(s) \right| \quad \left\| U_f \quad (s^0) \quad V_f \quad (s^0) \right\|_1 ;$$

which implies that:

$$\left\| (T \ U_f \quad)(s) \quad (T \ V_f \quad)(s) \right\|_1 \quad \left\| V_f \quad ^2(s^0) \quad V_f \quad ^2(s^0) \right\|_1 ;$$

Then based on the Contraction Mapping Theorem [23], we know that T has a unique fixed point $V_f(s); f \ 2 \ fr;cg$ such that $V_f(s) = (T \ V_f)(s)$. □

Proposition 2. *Suppose a trained policy ^{0} under the MC attacker satisfies: $V_c \ ^0 \ ^{MC}(\ _0)$, then ^{0} is guaranteed to be feasible with any B_p bounded adversarial perturbations.*

Similarly, suppose a trained policy ^{0} under the MR attacker satisfies: $V_c \ ^0 \ ^{MR}(\ _0)$, then ^{0} is guaranteed to be non-tempting with any B_p bounded adversarial perturbations.

Proposition 2 indicates that by solving the adversarial constrained optimization problem under the MC attacker, all the feasible solutions will be safe under any bounded adversarial perturbations. It also shows a nice property for training a robust policy, since the `max` operation over the reward in the safe RL objective may lead the policy to the tempting policy class, while the adversarial training with MR attacker can naturally keep the trained policy at a safe distance from the tempting policy class. Practically, we observe that both MC and MR attackers can increase the robustness and safety via adversarial training, and could be easily plugged into any on-policy safe RL algorithms, in principle. We leave the robust training framework for off-policy safe RL methods as future work.

Before proving Proposition 2, we first give the following definitions and lemmas.

Definition 1. Define the Bellman adversary effectiveness operator as $T_c : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$:

$$(T_c V_c)(s) = \max_{\bar{s} \in B_p(s)} \sum_{a \in A} (a|\bar{s}) \sum_{s' \in S} p(s'|s,a) [c(s,a,s') + V_c(s')]. \tag{51}$$

Definition 2. Define the Bellman adversary reward stealthiness operator as $T_r : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$:

$$(T_r V_r)(s) = \max_{\bar{s} \in B_p(s)} \sum_{a \in A} (a|\bar{s}) \sum_{s' \in S} p(s'|s,a) [r(s,a,s') + V_r(s')]. \tag{52}$$

Recall that $B_p(s)$ is the ℓ_p ball to constrain the perturbation range. The two definitions correspond to computing the value of the most effective and the most reward-stealthy attackers, which is similar to the Bellman optimality operator in the literature. We then show their contraction properties via the following Lemma:

Lemma 3. *The Bellman operators T_c; T_r are contractions under the sup-norm $\|\cdot\|_1$ and will converge to their fixed points, respectively. The fixed point for T_c is $V_c^{MC} = T_c V_c^{MC}$, and the fixed point for T_r is $V_r^{MR} = T_r V_r^{MR}$.*

To finish the proof of Lemma 3, we introduce another lemma:

Lemma 4. *Suppose $\max_x h(x) - \max_x g(x)$ and denote $x^h = \arg\max_x h(x)$, we have:*

$$\begin{aligned} |\max_x h(x) - \max_x g(x)| &= \max_x h(x) - \max_x g(x) = h(x^h) - \max_x g(x) \\ &\le h(x^h) - g(x^h) \le \max_x |h(x) - g(x)|. \end{aligned} \tag{53}$$

We then prove the Bellman contraction properties of Lemma 3:

Proof.

$$\left|(T_f V_f^{(1)})(s) - (T_f V_f^{(2)})(s)\right| = \left|\max_{\bar{s} \in B_p(s)} \sum_{a \in A} (a|\bar{s}) \sum_{s' \in S} p_{sa}^{s'} \left[f_{sa}^{s'} + V_f^{(1)}(s')\right]\right. \tag{54}$$

$$\left. - \max_{\bar{s} \in B_p(s)} \sum_{a \in A} (a|\bar{s}) \sum_{s' \in S} p_{sa}^{s'} \left[f_{sa}^{s'} + V_f^{(2)}(s')\right]\right| \tag{55}$$

$$= \left|\max_{\bar{s} \in B_p(s)} \sum_{a \in A} (a|\bar{s}) \sum_{s' \in S} p_{sa}^{s'} \left[V_f^{(1)}(s') - V_f^{(2)}(s')\right]\right| \tag{56}$$

$$\max_{\bar{s} \in B_p(s)} \left|\sum_{a \in A} (a|\bar{s}) \sum_{s' \in S} p_{sa}^{s'} \left[V_f^{(1)}(s') - V_f^{(2)}(s')\right]\right| \tag{57}$$

$$\triangleq \left|\sum_{a \in A} (a|\bar{s}^*) \sum_{s' \in S} p_{sa}^{s'} \left[V_f^{(1)}(s') - V_f^{(2)}(s')\right]\right| \tag{58}$$

$$\max_{s' \in S} \left|V_f^{(1)}(s') - V_f^{(2)}(s')\right| \tag{59}$$

$$= \left\|V_f^{(1)}(s') - V_f^{(2)}(s')\right\|_1; \tag{60}$$

where inequality (57) comes from Lemma 4, and $\bar{s}^*$ in Eq. (58) denote the argmax of the RHS.

Since the above holds for any state s, we can also conclude that:

$$\left\|(T_f V_f^{(1)})(s) - (T_f V_f^{(2)})(s)\right\|_1 \le \left\|V_f^{(1)}(s') - V_f^{(2)}(s')\right\|_1;$$

□

After proving the contraction, we prove that the value function of the MC and MR adversaries $V_{c\ ^{MC}}(s); V_{r\ ^{MR}}(s)$ are the fixed points for T_c ; T_r as follows:

Proof. Recall that the MC, MR adversaries are:

$$_{MC}(s) = \arg\max_{\tilde{s}2B_p(s)} \mathbb{E}_{\tilde{a}\ (a|\tilde{s})}[Q_c(s,a))]; \quad _{MR}(s) = \arg\max_{\tilde{s}2B_p(s)} \mathbb{E}_{\tilde{a}\ (a|\tilde{s})}[Q_r(s,a))]: \tag{61}$$

Based on the value function definition, we have:

$$V_{c\ ^{MC}}(s) = \mathbb{E}_{_{MC};s_0=s}[\sum_{t=0}^{1} {}^t c_t] = \mathbb{E}_{_{MC};s_0=s}[c_0 + \sum_{t=1}^{1} {}^{t\ 1} c_t] \tag{62}$$

$$= \sum_{a2A} (a|\ _{MC}(s)) \sum_{s'2S} p^{s'}_{sa} \left[c(s,a,s') + \mathbb{E}_{_{MC};s_1=s'}[\sum_{t=1}^{1} {}^{t\ 1} c_t] \right] \tag{63}$$

$$= \sum_{a2A} (a|\ _{MC}(s)) \sum_{s'2S} p^{s'}_{sa} [c(s,a,s') + V_{c\ ^{MC}}(s')] \tag{64}$$

$$= \max_{\tilde{s}2B_p(s)} \sum_{a2A} (a|\tilde{s}) \sum_{s'2S} p^{s'}_{sa} [c(s,a,s') + V_{c\ ^{MC}}(s')] \tag{65}$$

$$= (T_c\ V_{c\ ^{MC}})(s); \tag{66}$$

where Eq. (65) is from the MC attacker definition. Therefore, the cost value function of the MC attacker $V_{c\ ^{MC}}$ is the fixed point of the Bellman adversary effectiveness operator T_c . With the same procedure (replacing $_{MC}; T_c$ with $_{MR}; T_r$), we can prove that the reward value function of the MR attacker $V_{r\ ^{MR}}$ is the fixed point of the Bellman adversary stealthiness operator T_r .

□

With Lemma 3 and the proof above, we can easily obtain the conclusions in Proposition 2: if the trained policy is safe under the MC or the MR attacker, then it is guaranteed to be feasible or non-tempting under any $B_p(s)$ bounded adversarial perturbations respectively, since there are no other attackers can achieve higher cost or reward returns than them.

B Remarks

B.1 Remarks of the safe RL setting, stealthiness, and assumptions

Safe RL setting regarding the reward and the cost. We consider the safe RL problems that have separate task rewards and constraint violation costs, i.e. independent reward and cost functions. Combining the cost with reward to a single scalar metric, which can be viewed as manually selecting Lagrange multipliers, may work in simple problems. However, it lacks interpretability – it is hard to explain what does a single scalar value mean, and requires good domain knowledge of the problem – the weight between costs and rewards should be carefully balanced, which is difficult when the task rewards already contain many objectives/factors. On the other hand, separating the costs from rewards is easy to monitor the safety performance and task performance respectively, which is more interpretable and applicable for different cost constraint thresholds.

Safe RL setting regarding the constraint violations. There are multiple definitions of the safe RL problem, such as safe deployment and safe exploration. *Safe deployment* focuses on the safety of the policy *after* training, so constraint violations during training are tolerant. This is a widely used setting in many constrained-optimization-based safe RL papers, since it requires minimum domain knowledge of the problem and could be applied to a wide range of tasks. *Safe exploration* requires the agent to be safe throughout the training and testing time, which is much stricter than safe deployment. However, without domain knowledge or certain assumptions of the tasks, safe exploration is very difficult to solve. While we focus on the safe deployment setting in this paper, we want to point out that safe exploration is also an interesting but also challenging direction.

(Reward) Stealthy attack for safe RL. As we discussed in Sec. 3.2, the stealthiness concept in supervised learning refers to that the adversarial attack should be covert to prevent from being easily identified. While we use the perturbation set B_p to ensure the stealthiness regarding the observation corruption, we notice that another level of stealthiness regarding the task reward performance is interesting and worthy of being discussed. In some real-world applications, the task-related metrics (such as velocity, acceleration, goal distances) are usually easy to be monitored from sensors. However, the safety metrics can be sparse and hard to monitor until breaking the constraints, such as colliding with obstacles and entering hazard states, which are determined by binary indicator signals. Therefore, a dramatic task-related metrics (reward) drop might be easily detected by the agent, while constraint violation signals could be hard to detect until catastrophic failures. An unstealthy attack in this scenario may decrease the reward a lot and prohibit the agent from finishing the task, which can warn the agent that it is attacked and thus lead to a failing attack. On the contrary, a stealthy attack can maintain the agent's task reward such that the agent is not aware of the existence of the attacks based on "good" task metrics, while performing successful attacks by leading to constraint violations. In other words, a stealthy attack should corrupt the policy to be tempted, since all the tempting policies are high-rewarding while unsafe.

Stealthiness definition of the attacks. There is an alternative definition of stealthiness by viewing the difference in the reward regardless of increasing or decreasing. The two-sided stealthiness is a more strict one than the one-sided lower-bound definition in this paper. However, if we consider a practical system design, people usually set a threshold for the lower bound of the task performance to determine whether the system functions properly, rather than specifying an upper bound of the performance because it might be tricky to determine what should be the upper-bound of the task performance to be alerted by the agent. For instance, an autonomous vehicle that fails to reach the destination within a certain amount of time may be identified as abnormal, while reaching the goal faster may not since it might be hard to specify such a threshold to determine what is an overly good performance. Therefore, increasing the reward with the same amount of decreasing it may not attract the same attention from the agents. In addition, finding a stealthy and effective attacker with minimum reward change might be a much harder problem with the two-sided definition, since the candidate solutions are much fewer and the optimization problem could be harder to be formulated. But we believe that this is an interesting point that is worthy to be investigated in the future, while we will focus on the one-sided definition of stealthiness in this work.

Independently estimated reward and cost value functions assumption. Similar to most existing safe RL algorithms, such as PPO-Lagrangian [2, 3], CPO [17], FOCOPS [19], and CVPO [5], we consider the policy-based (or actor-critic-based) safe RL in this work. There are two phases for this type of approach: policy evaluation and policy improvement. In the **policy evaluation** phase, the

reward and cost value functions V_r ; V_c are evaluated separately. At this stage, the Bellman operators for reward and cost values are *independent*. Therefore, they have contractions (Def. 6) and will converge to their fixed points separately. In the **policy improvement** phase, the interconnections between the cost and reward functions exist since the policy optimizer needs to balance the two value functions together. However, no Bellman operators are involved in this phase. Therefore, we avoid the 'combined iteration' for computing the reward and cost values. This is a commonly used treatment in safe RL papers to train the policy: first evaluating the reward and cost values independently by Bellman equations and then optimizing the policy based on the learned value estimations. Therefore, our theoretical analysis of the Bellman contraction properties is also developed under this setting.

Locally Lipschitz assumption of the policy class. We use the locally Lipschitzness assumption for deriving Theorem 2 and 3. It is a reasonable assumption from a theoretical point of view because it is necessary for analysis. Without this assumption, the attacker may arbitrarily change the cost values, and it is hard to conduct a useful theoretical analysis. For the same reason, the Lipschitzness assumption is widely used in the adversarial ML and the RL literature, explicitly or implicitly. Second, we want to emphasize that the assumption is used in analysis rather than implementation. Even if a policy is not locally Lipschitz, the attackers and the adversarial training framework can still be implemented by using black-box optimizers to obtain the adversarial perturbations. Finally, in the real world, actions usually take effects via physical actuators that are usually continuous. For example, a real-world bang-bang controller for MountainCar sends signals to the engine or hydraulic braking systems. Their effects on states are continuous with finite Lipschitz constant. Therefore, the assumption is often valid in robotics.

Non-tempting safe RL problem. According to Def. 1-3, no tempting policy indicates a non-tempting safe RL problem, where the optimal policy has the highest reward while remaining under the constraint threshold. In the non-tempting setting, the MR attacker is not effective anymore, since maximizing the reward can not lead to tempting policies and constraint violations, while the MC attacker could still be effective. However, for the safe deployment problem that only cares about safety after training, no tempting policy means that the cost signal is unnecessary for training, because one can simply focus on maximizing the reward. As long as the most rewarding policies are found, the safety requirement would be automatically satisfied, and thus many standard RL algorithms can solve the problem. Since safe RL methods are not required in this setting, the non-tempting tasks are usually not discussed in safe RL papers, and are also not the focus of this paper.

B.2 Remarks of the failure of SA-PPOL(MC/MR) baselines

The detailed algorithm of SA-PPOL [11] can be found in Appendix C.5. The basic idea can be summarized via the following equation:

$$\grave{} \quad (s) = \quad D_{KL}[\ (\ js)jj\quad (\ j\ (s))]; \tag{67}$$

which aims to minimize the divergence between the corrupted and original states. Note that we only optimize (compute gradient) for $(\ j\ (s))$ rather than $(\ js)$, since we view $(\ js)$ as the "ground-truth" target action distribution. Adding the above KL regularizer to the original PPOL loss yields the SA-PPOL algorithm. We could observe the original SA-PPOL that uses the MAD attacker as the adversary can learn well in most of the tasks, though it is not safe under strong attacks. However, SA-PPOL with MR or MC adversaries often fail to learn a meaningful policy in many tasks, especially for the MR attacker. The reason is that: the MR attacker aims to find the high-rewarding adversarial states, while the KL loss will make the policy distribution of high-rewarding adversarial states to match with the policy distribution of the original relatively lower-rewards states. As a result, the training could fail due to wrong policy optimization direction and prohibited exploration to high-rewarding states. Since the MC attacker can also lead to high-rewarding adversarial states due to the existence of tempting polices, we may also observe failure training with the MC attacker.

C Implementation Details

C.1 MC and MR attackers implementation

Practically, given a fixed policy and its critics $Q_f(s,a); f \in \{r,c\}$, we obtain the corrupted state $\tilde{s}$ of s from the MR and MC attackers by solving:

$$\nu_{MR}(s) = \arg\max_{\tilde{s}\in B_p(s)} \mathbb{E}_{\tilde{a}\sim(a|\tilde{s})}[Q_r(s,a))]; \quad \nu_{MC}(s) = \arg\max_{\tilde{s}\in B_p(s)} \mathbb{E}_{\tilde{a}\sim(a|\tilde{s})}[Q_c(s,a))] \tag{68}$$

Suppose the policy and the critics Q are all parametrized by differentiable models such as neural networks, then we can back-propagate the gradient through Q and to solve the adversarial state $\tilde{s}$. This is similar to the policy optimization procedure in TD3 [24] and DDPG [25], whereas we replace the optimization domain from the policy parameter space to the observation space $B_p(s)$.

We use the gradient of the state-action value function $Q(s,a)$ to provide the direction to update states adversarially in K steps ($Q = Q_r$ for MR and $Q = Q_c$ for MC):

$$s^{k+1} = \text{Proj}[s^k \quad r_{s^k} Q(s^0, \quad (s^k))]; k = 0,\dots,K \quad 1 \tag{69}$$

where Proj[ ] is a projection to $B_p(s^0)$, is the learning rate, and s^0 is the state under attack. Note that we use the gradient of $Q(s^0, \quad (s^k))$ rather than $Q(s^k, \quad (s^k))$ to make the optimization more stable, since the Q function may not generalize well to unseen states in practice. The implementation of MC and MR attacker is shown in algorithm 1.

Algorithm 1 MC and MR attacker

Input: A policy under attack, corresponding Q networks, initial state s^0, attack steps K, attacker learning rate , perturbation range , two thresholds $_Q$ and $_s$ for early stopping
Output: An adversarial state $\tilde{s}$

```
for k = 1 to K do
    g^k = r_{s^k 1} Q(s_0, (s^k 1))
    s^k  Proj[s^k 1  g^k]
    Compute Q = jQ(s_0, (s^k))  Q(s_0, (s^k 1))j and s = js^k  s^k 1j
    if Q <  _Q and s <  _s then
        break for early stopping
    end if
end for
```

C.2 PPO-Lagrangian algorithm

The objective of PPO (clipped) has the form [26]:

$$\grave{}_{ppo} = \min(\frac{(a|s)}{_k(a|s)} A^{\ k}(s,a); \text{clip}(\frac{(a|s)}{_k(a|s)}; 1 \quad ; 1 + \ )A^{\ k}(s,a)) \tag{70}$$

We use PID Lagrangian [3] that addresses the oscillation and overshoot problem in Lagrangian methods. The loss of the PPO-Lagrangian has the form:

$$\grave{}_{ppol} = \frac{1}{1+}(\grave{}_{ppo} + V_r \quad V_c) \tag{71}$$

The Lagrangian multiplier is computed by applying feedback control to V_c and is determined by K_P, K_I, and K_D that need to be fine-tuned.

C.3 Adversarial training practical implementation

Algorithm 2 Adversarial safe RL training meta algorithm

Input: Safe RL `learner`, Adversary `scheduler`
Output: Observational robust policy

1: Initialize policy $\quad 2 \quad$ and adversary $\quad : S \; ! \; S$
2: **for** each training epoch $n = 1; :::; N$ **do**
3: Rollout trajectories: $\sim = fs_0; a_0; :::g_T, a_t \qquad (aj \;\; (s_t))$
4: Run safe RL learner: `learner`$(\sim; \quad)$
5: Update adversary: `scheduler`$(\sim; \quad ; n)$
6: **end for**

Due to the page limit, we omit some implementation details in the main content. The meta adversarial training algorithm is shown in Algo. 2. We particularly adopt the primal-dual methods [2, 3] that are widely used in the safe RL literature as the `learner`, then the adversarial training objective can be converted to a min-max form by using the Lagrange multiplier $\quad$:

$$(\quad ; \quad) = \min_{0} \max_{2\Pi} V_r \quad (\quad _0) \qquad (V_c \quad (\quad _0) \qquad) \tag{72}$$

Solving the inner maximization (primal update) via any policy optimization methods and the outer minimization (dual update) via gradient descent iteratively yields the Lagrangian algorithm. Under proper learning rates and bounded noise assumptions, the iterates $(\quad _n; \quad _n)$ converge to a fixed point (a local minimum) almost surely [27, 28].

Based on previous theoretical analysis, we adopt MC or MR as the adversary when sampling trajectories. The `scheduler` function aims to train the reward and cost Q-value functions for the MR and the MC attackers, because many on-policy algorithms such as PPO do not provide them. In addition, the scheduler can update the adversary's power based on the learning progress accordingly, since a strong adversary at the beginning may prohibit the `learner` from exploring the environment and thus corrupt the training. We gradually increase the perturbation range $\quad$ along with the training epochs to adjust the adversary perturbation set B_p, such that the agent will not be too conservative in the early stage of training. A similar idea is also used in adversarial training [29–31] and curriculum learning literature [32, 33].

We then present the full algorithm and some implementation tricks in the following. Without otherwise statements, the critics' and policies' parameterization is assumed to be neural networks (NN), while we believe other parameterization forms should also work well.

Critics update. Denote $\quad_r$ as the parameters for the task reward critic Q_r, and $\quad_c$ as the parameters for the constraint violation cost critic Q_c. Similar to many other off-policy algorithms [25], we use a target network for each critic and the polyak smoothing trick to stabilize the training. Other off-policy critics training methods, such as Re-trace [34], could also be easily incorporated with PPO-Lagrangian training framework. Denote $\quad^0_r$ as the parameters for the **target** reward critic Q^0_r, and $\quad^0_c$ as the parameters for the **target** cost critic Q^0_c. Define D as the replay buffer and $(s; a; s^0; r; c)$ as the state, action, next state, reward, and cost respectively. The critics are updated by minimizing the following mean-squared Bellman error (MSBE):

$$`(\quad _r) = \mathsf{E}_{(s;a;s^0;r;c) \quad D} \left[(Q_r(s; a) \quad (r + \quad \mathsf{E}_{a^0} \quad [Q^0_r(s^0; a^0)]))^2 \right] \tag{73}$$

$$`(\quad _c) = \mathsf{E}_{(s;a;s^0;r;c) \quad D} \left[(Q_c(s; a) \quad (c + \quad \mathsf{E}_{a^0} \quad [Q^0_c(s^0; a^0)]))^2 \right]: \tag{74}$$

Denote $\quad_c$ as the critics' learning rate, we have the following updating equations:

$$_r \qquad _r \qquad _c r \quad _r `(\quad _r) \tag{75}$$

$$_c \qquad _c \qquad _c r \quad _c `(\quad _c) \tag{76}$$

Note that the original PPO-Lagrangian algorithm is an on-policy algorithm, which doesn't require the reward critic and cost critic to train the policy. We learn the critics because the MC and MR attackers require them, which is an essential module for adversarial training.

Polyak averaging for the target networks. The polyak averaging is specified by a weight parameter $\in (0, 1)$ and updates the parameters with:

$$\begin{aligned} {}^{0}_{r} &= {}^{0}_{r} + (1 -) \ _{r} \\ {}^{0}_{c} &= {}^{0}_{c} + (1 -) \ _{c} \\ {}^{0} &= {}^{0} + (1 -) \ . \end{aligned} \tag{77}$$

The critic's training tricks are widely adopted in many off-policy RL algorithms, such as SAC, DDPG and TD3. We observe that the critics trained with those implementation tricks work well in practice. Then we present the full Robust PPO-Lagrangian algorithm:

Algorithm 3 Robust PPO-Lagrangian Algorithm

Input: rollouts T, policy optimization steps M, PPO-Lag loss function $\grave{}_{ppol}(s, \ , r, c)$, adversary function (s), policy parameter $\ $, critic parameter $_r$ and $_c$, target critic parameter ${}^{0}_{r}$ and ${}^{0}_{c}$
Output: policy

```
Initialize policy parameters and critics parameters
for each training iteration do
    Rollout T trajectories by  from the environment f( (s); (a); (s'); r; c)g_N
    . Update learner
    for Optimization steps m = 1; :::; M do
        . No KL regularizer!
        Compute PPO-Lag loss `_ppol(s; ; r; c) by Eq. (71)
        Update actor  r `_ppo
    end for
    Update value function based on samples f(s; a; s'; r; c)g_N
    . Update adversary
    Update critics Q_c and Q_r by Eq. (75) and Eq. (76)
    Polyak averaging target networks by Eq. (77)
    Update adversary based on Q_c and Q_r
end for
```

C.4 Attacker baselines

Maximum Action Difference (MAD) attacker baseline. The MAD attacker is designed for standard RL tasks [11], which is shown to be effective in decreasing a trained RL agent's reward return; The optimal adversarial observation is obtained by maximizing the KL-divergence between the corrupted policy: $_{\text{MAD}}(s) = \arg\max_{\tilde{s} \in B_p(s)} D_{\text{KL}}[\ (a|s) \ k \ (a|\tilde{s})]$

Adaptive MAD (AMAD) attacker baseline. Since the vanilla MAD attacker is not designed for safe RL, we further improve it to an adaptive version as a stronger baseline. The motivation comes from Lemma 2 – the optimal policy will be close to the constraint boundary that with high risks. To better understand this property, we introduce the discounted future state distribution $d \ (s)$ [35], which allows us to rewrite the result in Lemma 2 as (see Appendix C.4 for derivation and implementation details): $\frac{1}{1-} \int_{s \in S} d \ (s) \int_{a \in A} \ (a|s) \int_{s' \in S} p(s'|s, a) c(s, a, s') ds' da ds = \ $. We can see that performing MAD attack for the optimal policy in low-risk regions that with small $p(s'|s, a)c(s, a, s')$ values may not be effective. Therefore, AMAD only perturbs the observation when the agent is within high-risk regions that are determined by the cost value function and a threshold to achieve more effective attacks: $_{\text{AMAD}}(s) := \begin{cases} _{\text{MAD}}(s), & \text{if } V_c \ (s) \ \ ; \\ s, & \text{otherwise} . \end{cases}$

The full algorithm of MAD attacker is presented in algorithm 4. We use the same SGLD optimizer as in [11] to maximize the KL-divergence. The objective of the MAD attacker is defined as:

$$\grave{}_{MAD}(s) = \ D_{KL}[\ (\cdot|s_0) \| \ (\cdot|s)] \tag{78}$$

Note that we back-propagate the gradient from the corrupted state s instead of the original state s_0 to the policy parameters $\ $. The full algorithm is shown below:

Algorithm 4 MAD attacker

Input: A policy under attack, corresponding $Q(s;a)$ network, initial state s^0, attack steps K, attacker learning rate , the (inverse) temperature parameter for SGLD , two thresholds $_Q$ and $_s$ for early stopping
Output: An adversarial state $\tilde{s}$

1: **for** $k = 1$ to K **do**
2: Sample $N(0;1)$
3: $g^k = r\,\tilde{}_{MAD}(s_{t\ 1}) + \sqrt{\frac{2}{}}$
4: s^k Proj$[s^{k\ 1}$ $g^k]$
5: Compute $Q = jQ(s_0;\ (s^k))$ $Q(s_0;\ (s^{k\ 1}))j$ and $s = js^k$ $s^{k\ 1}j$
6: **if** $Q <$ $_Q$ and $s <$ $_s$ **then**
7: break for early stopping
8: **end if**
9: **end for**

To motivate the design of AMAD baseline, we denote $P\ (s^0js) = \int p(s^0js;a)\ (ajs)da$ as the state transition kernel and $p_t\ (s) = p(s_t = sj\ )$ as the probability of visiting the state s at the time t under the policy , where $p_t\ (s^0) = \int P\ (s^0js)p_{t\ 1}(s)ds$. Then the discounted future state distribution $d\ (s)$ is defined as [35]:

$$d\ (s) = (1\quad)\sum_{t=0}^{1}\ {}^t p_t\ (s);$$

which allows us to represent the value functions compactly:

$$\begin{aligned} V_f\ (\ _0) &= \frac{1}{1}\mathsf{E}_{s\ \ d\ ;a\ \ ;s^0\ \ p}[f\ (s;a;s^0)] \\ &= \frac{1}{1}\int_{s2S} d\ (s)\int_{a2A}\ (ajs)\int_{s^02S} p(s^0js;a)f\ (s;a;s^0)ds^0dads; \quad f\ 2\ fr;cg \end{aligned} \tag{79}$$

Based on Lemma 2, the optimal policy in a tempting safe RL setting satisfies:

$$\frac{1}{1}\int_{s2S} d\ \ (s)\int_{a2A}\ \ (ajs)\int_{s^02S} p(s^0js;a)c(s;a;s^0)ds^0dads = \ : \tag{80}$$

We can see that performing MAD attack in low-risk regions that with small $p(s^0js;a)c(s;a;s^0)$ values may not be effective – the agent may not even be close to the safety boundary. On the other hand, perturbing when $p(s^0js;a)c(s;a;s^0)$ is large may have higher chance to result in constraint violations. Therefore, we improve the MAD to the Adaptive MAD attacker, which will only attack the agent in high-risk regions (determined by the cost value function and a threshold).

The implementation of AMAD is shown in algorithm 5. Given a batch of states fsg_N, we compute the cost values $fV_c\ (s)g_N$ and sort them in ascending order. Then we select certain percentile of $fV_c\ (s)g_N$ as the threshold and attack the states that have higher cost value than .

Algorithm 5 AMAD attacker

Input: a batch of states fsg_N, threshold , a policy under attack, corresponding $Q(s;a)$ network, initial state s^0, attack steps K, attacker learning rate , the (inverse) temperature parameter for SGLD , two thresholds $_Q$ and $_s$ for early stopping
Output: batch adversarial state $\tilde{s}$

1: Compute batch cost values $fV_c\ (s)g_N$
2: (1) percentile of $V_c\ (s)$
3: **for** the state s that $V_c\ (s) >$ **do**
4: compute adversarial state $\tilde{s}$ by algorithm 4
5: **end for**

C.5 SA-PPO-Lagrangian baseline

Algorithm 6 SA-PPO-Lagrangian Algorithm

Input: rollouts T, policy optimization steps M, PPO-Lag loss function $\ell_{ppo}(s, \ , r, c)$, adversary function (s)
Output: policy

1: Initialize policy parameters and critics parameters
2: **for** each training iteration **do**
3: Rollout T trajectories by from the environment $\{(s, a, s', r, c)\}_N$
4: Compute adversary states $\tilde{s} = (s)$ for the sampled trajectories
5: *. Update actors*
6: **for** Optimization steps $m = 1, \ldots, M$ **do**
7: Compute KL robustness regularizer $L_{KL} = D_{\mathrm{KL}}((s) \| (\tilde{s}))$, no gradient from (s)
8: Compute PPO-Lag loss $\ell_{ppol}(s, \ , r, c)$ by Eq. (71)
9: Combine them together with a weight : $\ell = \ell_{ppol}(s, \ , r, c) + \ _{KL}$
10: Update actor $r \ \ell$
11: **end for**
12: *. Update critics*
13: Update value function based on samples $\{(s, a, s', r, c)\}_N$
14: **end for**

The SA-PPO-Lagrangian algorithm adds an additional KL robustness regularizer to robustify the training policy. Choosing different adversaries yields different baseline algorithms. The original SA-PPOL [11] method adopts the MAD attacker, while we conduct ablation studies by using the MR attacker and the MC attacker, which yields the SA-PPOL(MR) and the SA-PPOL(MC) baselines respectively.

C.6 Environment description

The simulation environments are from a public available benchmark [20]. We consider two tasks, and train multiple different robots (Car, Drone, Ant) for each task:

Run task. Agents are rewarded for running fast between two safety boundaries and are given costs for violation constraints if they run across the boundaries or exceed an agent-specific velocity threshold. The tempting policies can violate the velocity constraint to obtain more rewards.

Circle task. The agents are rewarded for running in a circle in a clockwise direction but are constrained to stay within a safe region that is smaller than the radius of the target circle. The tempting policies in this task will leave the safe region to gain more rewards.

We name each task via the `Robot-Task` format, for instance, `Car-Run`. More detailed descriptions and video demos are available on our anonymous project website [2].

In the Circle tasks, the goal is for an agent to move along the circumference of a circle while remaining within a safety region smaller than the radius of the circle. The reward and cost functions are defined as:

$$r(s) = \frac{yv_x + xv_y}{1 + |\sqrt{x^2 + y^2} \quad r|} + r_{robot}(s)$$
$$c(s) = \mathbf{1}(|x| > x_{lim})$$

where x, y are the position of the agent on the plane, v_x, v_y are the velocities of the agent along the x and y directions, r is the radius of the circle, and x_{lim} specified the range of the safety region, $r_{robot}(s)$ is the specific reward for different robot. For example, an ant robot will gain reward if its feet do not collide with each other. In the Run tasks, the goal for an agent is to move as far as possible

[2] `https://sites.google.com/view/robustsaferl/home`

within the safety region and the speed limit. The reward and cost functions are defined as:

$$r(s) = \sqrt{(x_{t\ 1} \quad g_x)^2 + (y_{t\ 1} \quad g_y)^2} \quad \sqrt{(x_t \quad g_x)^2 + (y_t \quad g_y)^2} + r_{robot}(s)$$
$$c(s) = \mathbf{1}(jyj > y_{lim}) + \mathbf{1}(\sqrt{v_x^2 + v_y^2} > v_{lim})$$

where v_{lim} is the speed limit and g_x and g_y is the position of a fictitious target. The reward is the difference between current distance to the target and the distance in the last timestamp.

C.7 Hyper-parameters

In all experiments, we use Gaussian policies with mean vectors given as the outputs of neural networks, and with variances that are separate learnable parameters. The policy networks and Q networks for all experiments have two hidden layers of sizes (256, 256) with ReLU activation functions. We use a discount factor of = 0:995, a GAE- for estimating the regular advantages of ^{GAE} = 0:97, a KL-divergence step size of $_{KL}$ = 0:01, a clipping coefficient of 0.02. The PID parameters for the Lagrange multiplier are: K_p = 0:1, K_I = 0:003, and K_D = 0:001. The learning rate of the adversarial attackers: MAD, AMAD, MC, and MR is 0.05. The optimization steps of MAD and AMAD is 60 and 200 for MC and MR attacker. The threshold for AMAD is 0.1. The complete hyperparameters used in the experiments are shown in Table 2. We choose larger perturbation range for the Car robot-related tasks because they are simpler and easier to train.

Table 2: Hyperparameters for all the environments

Parameter	Car-Run	Dron-Run	Ant-Run	Car-Circle	Dron-Circle	Ant-Circle
training epoch	100	250	250	500	500	1000
batch size	10000	20000	20000	15000	15000	30000
minibatch size	300	300	300	300	300	300
rollout length	200	100	200	300	300	300
cost limit	5	5	5	5	5	5
perturbation	0.05	0.025	0.025	0.05	0.025	0.025
actor optimization step M	80	80	80	80	80	160
actor learning rate	0.0003	0.0002	0.0005	0.0003	0.0003	0.0005
critic learning rate	0.001	0.001	0.001	0.001	0.001	0.001

C.8 More experiment results

All the experiments are performed on a server with AMD EPYC 7713 64-Core Processor CPU. For each experiment, we use 4 CPUs to train each agent that is implemented by PyTorch, and the training time varies from 4 hours (Car-Run) to 3 days (Ant-Circle). Video demos are available at: `https://sites.google.com/view/robustsaferl/home`

Here we evaluate the performance of MAD and AMAD adversaries by attacking well-trained PPO-Lagrangian policies in Car-Run and Ant-Run task. We keep the policies' model weights fixed for all the attackers for fair comparison. The comparison is shown in Fig. 3. We vary the attacking fraction (determined by) to thoroughly study the effectiveness of the AMAD attacker. We can see that AMAD attacker is more effective because the cost increases significantly with the increase in perturbation, while the reward is maintained well. This validates our hypothesis that attacking the agent in high-risk regions is more effective and stealthy.

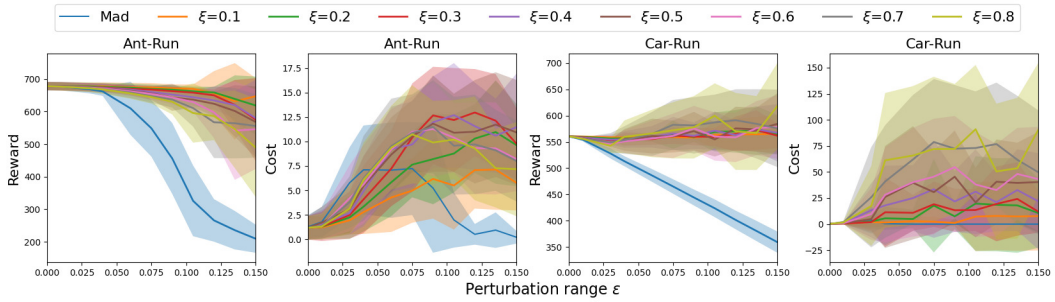


Figure 3: Reward and cost of AMAD and MAD attacker

The evaluation results of different trained policies under adversarial attacks are shown in Table 3 and the experiment results of trained safe RL policies under the Random and MAD attackers are shown in Table 4. The last column shows the average rewards and costs over all the 5 attackers (Random, MAD, AMAD, MC, MR). Our agent (ADV-PPOL) with adversarial training is robust against all the 5 attackers and achieves the lowest cost. We can also see that AMAD attacker is more effective than MAD since the cost under the AMAD attacker is higher than the cost under the MAD attacker.

Table 3: Evaluation results of natural performance (no attack) and under 3 attackers. Our methods are ADV-PPOL(MC/MR). Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds. We shadow two lowest-costs agents under each attacker column and break ties based on rewards, excluding the failing agents (whose natural rewards are less than 50% of PPOL-vanilla's). We mark the failing agents with ?.

Env	Method	Natural		AMAD		MC		MR	
		Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Car-Run = 0:05	PPOL-vanilla	561.33 1.97	0.15 0.36	548.04 17.72	13.49 30.93	624.49 8.87	184.09 0.52	624.46 8.86	184.06 0.47
	PPOL-random	556.84 1.87	0.01 0.08	550.66 1.71	1.73 2.05	584.18 2.69	183.79 0.73	585.79 2.01	183.91 0.59
	SA-PPOL	545.87 2.13	0.0 0.0	546.95 2.11	0.0 0.0	571.14 1.5	6.44 7.28	571.07 1.33	1.77 3.31
	SA-PPOL(MC)	552.58 3.84	0.0 0.0	541.58 3.6	0.0 0.0	568.96 1.92	1.17 2.07	569.6 1.51	0.67 1.24
	SA-PPOL(MR)	543.0 1.14	0.0 0.0	537.19 1.47	0.0 0.0	568.32 1.95	16.27 23.2	568.29 2.12	12.23 17.26
	ADV-PPOL(MC)	525.76 2.99	0.0 0.0	516.22 3.52	0.0 0.0	555.64 3.44	0.05 0.21	554.48 2.78	0.01 0.08
	ADV-PPOL(MR)	525.93 2.28	0.0 0.0	514.97 2.68	0.0 0.0	557.38 2.83	0.06 0.24	556.87 3.03	0.05 0.22
Drone-Run = 0:025	PPOL-vanilla	347.17 1.53	0.0 0.0	362.18 11.24	35.69 17.74	336.05 9.43	79.0 0.0	345.64 5.22	79.0 0.0
	PPOL-random	343.71 1.55	0.0 0.0	361.85 18.03	65.58 22.58	268.28 4.26	0.9 2.28	317.25 40.34	33.66 29.86
	SA-PPOL	284.47 32.13	0.0 0.0	306.55 26.44	11.85 18.04	156.97 384.11	61.88 18.49	403.11 163.75	75.73 7.71
	? SA-PPOL(MC)	174.61 34.37	0.06 0.73	86.35 56.22	0.0 0.0	205.34 29.57	10.13 10.67	217.51 24.77	8.31 6.05
	? SA-PPOL(MR)	0.13 0.22	0.0 0.0	0.11 0.21	0.0 0.0	0.25 0.37	0.0 0.0	0.28 0.43	0.0 0.0
	ADV-PPOL(MC)	273.4 16.98	0.0 0.0	268.0 12.0	0.05 0.57	275.0 28.26	1.1 3.17	294.8 23.67	18.11 25.87
	ADV-PPOL(MR)	233.31 20.68	0.0 0.0	238.0 22.15	0.0 0.0	229.8 68.0	6.81 8.38	238.11 46.17	0.95 1.92
Car Circle = 0:05	PPOL-vanilla	337.69 152.34	1.8 3.91	274.61 78.92	92.53 39.32	265.61 12.43	69.33 18.91	238.06 101.01	74.47 38.44
	PPOL-random	398.71 48.96	0.17 0.9	293.77 105.83	69.97 46.75	307.77 32.95	59.3 26.29	295.2 45.56	49.73 22.98
	SA-PPOL	403.92 22.63	0.4 2.15	382.8 20.57	0.37 1.97	361.0 12.96	109.1 6.0	452.98 26.72	89.03 9.13
	SA-PPOL(MC)	417.78 17.79	0.33 1.8	314.13 27.73	0.0 0.0	355.12 13.42	98.43 14.52	468.1 14.41	87.5 9.17
	SA-PPOL(MR)	389.03 47.53	0.2 1.0	351.49 34.16	0.14 0.69	342.67 39.23	77.9 21.57	414.87 66.09	75.68 20.73
	ADV-PPOL(MC)	302.3 12.24	0.1 0.7	296.23 19.02	1.86 5.49	310.37 25.68	1.12 3.98	261.52 24.51	0.28 1.59
	ADV-PPOL(MR)	309.42 35.45	0.0 0.0	321.44 20.52	6.66 10.94	258.52 31.53	0.08 0.56	308.6 54.7	0.16 1.12
Drone Circle = 0:025	PPOL-vanilla	627.49 55.24	0.27 1.12	527.6 171.54	34.5 36.73	228.79 181.92	95.17 59.23	85.79 159.67	174.4 81.58
	PPOL-random	604.31 46.83	0.37 1.97	559.2 173.25	27.67 32.66	159.16 184.15	91.5 98.26	130.08 146.67	103.1 92.03
	SA-PPOL	503.13 19.89	0.0 0.0	496.34 20.54	0.0 0.0	430.64 89.64	97.57 27.47	346.99 320.08	109.5 78.1
	SA-PPOL(MC)	347.43 97.49	8.5 35.32	346.25 41.68	0.0 0.0	329.05 143.47	58.77 34.94	380.53 176.19	78.07 60.05
	? SA-PPOL(MR)	184.7 128.7	11.94 43.67	189.76 118.14	15.38 47.38	189.18 142.46	44.62 35.83	219.87 138.35	49.14 52.87
	ADV-PPOL(MC)	359.02 33.01	0.0 0.0	351.57 52.5	1.48 6.44	399.78 69.47	4.16 12.57	356.09 90.42	9.66 28.48
	ADV-PPOL(MR)	356.6 46.91	0.0 0.0	339.04 72.43	5.36 23.08	275.43 95.08	5.66 22.41	379.52 87.22	1.2 6.47

Table 4: Evaluation results of natural performance (no attack) and under Random and MAD attackers. The average column shows the average rewards and costs over all 5 attackers (Random, MAD, AMAD, MC, and MR). Our methods are ADV-PPOL(MC/MR). Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds. We shadow two lowest-costs agents under each attacker column and break ties based on rewards, excluding the failing agents (whose natural rewards are less than 50% of PPOL-vanilla's. We mark the failing agents with ?.

Env	Method	Random		MAD		Average	
		Reward	Cost	Reward	Cost	Reward	Cost
Car-Circle = 0:05	PPOL-vanilla	328.34 118.08	20.67 18.65	178.21 81.31	28.27 49.12	280.42 87.0	42.41 13.74
	PPOL-random	393.49 43.87	2.17 3.44	272.0 75.21	81.63 43.47	337.16 51.76	37.65 12.91
	SA-PPOL(MAD)	397.71 20.87	0.13 0.5	366.64 25.82	0.93 2.73	394.92 15.33	28.57 1.75
	SA-PPOL(MC)	383.92 19.5	0.0 0.0	288.28 25.2	0.0 0.0	376.5 7.09	26.61 2.39
	SA-PPOL(MR)	370.71 47.18	0.54 1.94	308.8 22.49	0.1 0.7	366.16 37.08	22.08 4.6
	ADV-PPOL(MC)	302.61 11.81	0.0 0.0	292.83 23.04	2.22 4.75	295.46 8.04	0.83 1.35
	ADV-PPOL(MR)	309.81 34.96	0.0 0.0	312.18 15.81	8.76 11.52	304.34 10.29	2.24 3.1
Drone-Circle = 0:025	PPOL-vanilla	603.53 85.34	6.5 7.03	469.47 186.11	69.17 38.21	452.41 51.12	54.57 19.34
	PPOL-random	585.71 108.76	6.87 33.14	456.66 155.61	58.6 42.59	442.21 44.87	41.21 24.18
	SA-PPOL(MAD)	500.49 18.23	0.0 0.0	491.23 25.15	0.23 0.96	467.1 54.85	29.61 12.62
	SA-PPOL(MC)	357.65 49.52	0.0 0.0	343.52 50.41	0.47 1.77	352.13 49.02	20.83 10.19
	? SA-PPOL(MR)	187.81 129.74	19.18 53.92	180.62 122.42	15.06 41.66	191.66 123.83	23.4 21.63
	ADV-PPOL(MC)	359.45 26.63	0.0 0.0	325.92 46.12	4.22 13.82	358.74 35.95	2.79 4.97
	ADV-PPOL(MR)	352.77 51.5	0.0 0.0	331.06 63.45	4.4 14.96	341.8 37.13	2.37 6.06
Ant-Circle = 0:025	PPOL-vanilla	152.98 21.02	0.9 3.59	157.36 22.76	5.27 10.27	160.93 17.15	17.69 7.0
	PPOL-random	159.02 23.93	3.13 8.15	155.34 27.44	2.8 5.47	153.63 13.59	10.56 4.41
	SA-PPOL(MAD)	140.21 39.95	4.6 21.18	146.38 34.43	1.47 5.19	152.47 22.08	14.4 8.09
	? SA-PPOL(MC)	-0.38 1.57	0.0 0.0	-0.73 1.88	0.0 0.0	-0.3 0.8	0.0 0.0
	? SA-PPOL(MR)	-0.53 2.07	0.0 0.0	-0.89 2.25	0.0 0.0	-0.66 1.09	0.0 0.0
	ADV-PPOL(MC)	131.22 18.72	0.3 1.29	132.95 18.85	0.03 0.18	132.55 14.1	1.86 2.51
	ADV-PPOL(MR)	126.91 23.59	0.73 2.93	134.82 19.38	1.63 4.37	131.35 13.22	1.02 1.56
Car-Run = 0:05	PPOL-vanilla	553.61 2.81	19.47 6.19	504.29 9.71	0.49 5.94	567.75 3.38	58.84 4.68
	PPOL-random	555.24 1.89	0.92 1.17	542.84 2.2	2.61 2.44	561.68 1.52	53.28 0.49
	SA-PPOL(MAD)	545.86 2.11	0.0 0.0	548.11 2.2	0.0 0.0	553.52 1.62	1.17 1.31
	SA-PPOL(MC)	540.36 2.83	0.0 0.0	522.8 3.1	0.0 0.0	549.06 2.55	0.26 0.44
	SA-PPOL(MR)	539.04 1.31	0.0 0.0	529.38 1.91	0.0 0.0	546.74 1.12	4.07 5.73
	ADV-PPOL(MC)	521.85 3.2	0.0 0.0	504.25 4.23	0.0 0.0	528.93 2.74	0.01 0.04
	ADV-PPOL(MR)	522.15 2.31	0.0 0.0	504.16 3.12	0.0 0.0	529.41 2.36	0.02 0.05
Drone-Run = 0:025	PPOL-vanilla	346.59 2.93	17.33 12.63	348.19 33.21	41.96 28.11	347.52 5.44	37.31 5.46
	PPOL-random	342.68 3.16	3.72 5.6	269.88 14.33	1.66 8.4	321.03 8.03	15.11 6.72
	SA-PPOL(MAD)	306.73 20.71	1.9 4.8	323.19 24.66	29.19 23.81	296.86 53.06	25.79 5.5
	? SA-PPOL(MC)	151.69 19.97	0.01 0.16	77.66 49.01	0.0 0.0	155.15 11.8	2.67 2.11
	? SA-PPOL(MR)	0.09 0.29	0.0 0.0	0.05 0.28	0.0 0.0	0.15 0.29	0.0 0.0
	ADV-PPOL(MC)	277.23 6.89	0.0 0.0	264.26 12.98	0.12 0.69	275.86 9.33	2.77 3.78
	ADV-PPOL(MR)	235.17 20.24	0.0 0.0	230.04 24.4	0.0 0.0	233.9 25.83	1.11 1.29
Ant-Run = 0:025	PPOL-vanilla	676.14 12.12	1.89 1.77	672.38 12.71	3.59 2.66	678.37 13.99	27.09 5.09
	PPOL-random	671.8 14.45	1.52 1.2	667.73 13.6	2.09 1.43	669.79 8.59	14.37 2.12
	SA-PPOL(MAD)	659.01 13.66	0.55 0.8	658.28 13.9	0.75 0.98	665.21 9.41	22.52 4.88
	SA-PPOL(MC)	575.82 27.89	3.44 3.51	572.12 27.95	3.17 3.47	584.98 25.62	10.06 4.23
	? SA-PPOL(MR)	68.46 93.11	5.27 4.46	77.65 79.75	5.17 4.5	77.83 67.63	5.58 4.34
	ADV-PPOL(MC)	599.93 18.22	0.0 0.0	597.65 18.61	0.0 0.0	618.73 17.7	0.41 0.24
	ADV-PPOL(MR)	618.62 25.38	0.31 0.6	615.31 23.5	0.41 0.68	625.14 21.95	1.46 0.74

The experiment results of the maximum-entropy method: SAC-Lagrangian is shown in Table 5. We evaluated the effect of different entropy regularizers on the robustness against observational perturbation. Although the trained agents can achieve almost zero constraint violations in noise-free environments, they suffer from vulnerability issues under the proposed MC and MR attacks. Increasing the entropy cannot make the agent more robust against adversarial attacks.

Table 5: Evaluation results of natural performance (no attack) and under MC and MR attackers of SAC-Lagrangian w.r.t different entropy regularizer . Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds.

Env		Natural		MC		MR	
		Reward	Cost	Reward	Cost	Reward	Cost
Car-Circle = 0:05	0.1	414.43 7.99	1.04 2.07	342.32 17.8	112.53 6.92	328.5 22.06	43.52 18.39
	0.01	437.12 9.83	0.94 1.96	309.0 60.72	92.53 22.04	313.58 21.1	35.0 15.59
	0.001	437.41 10.0	1.15 2.36	261.1 53.0	65.92 24.37	383.09 50.06	53.92 16.3
	0.0001	369.79 130.6	5.23 11.13	276.32 107.11	84.21 35.68	347.85 117.79	52.97 22.4
Car-Run = 0:05	0.1	544.77 17.44	0.32 0.71	599.54 10.08	167.77 29.44	591.7 27.82	158.71 51.95
	0.01	521.12 23.42	0.19 0.5	549.43 53.71	73.31 54.43	535.99 23.34	21.71 24.25
	0.001	516.22 47.14	0.47 0.95	550.29 34.78	90.47 48.81	546.3 54.49	87.25 60.46
	0.0001	434.92 136.81	0.0 0.0	446.44 151.74	41.89 44.41	452.29 119.77	15.16 17.7

The experiment results of FOCOPS [19] is shown in Table 6. We trained FOCOPS without adversarial attackers FOCOPS-vanilla and with our adversarial training methods FOCOPS(MR) and FOCOPS(MR) under the MC and MR attackers respectively. We can see that the vanilla method is safe in noise-free environments, however, they are not safe anymore under the proposed adversarial attack. In addition, the adversarial training can help to improve the robustness and make the FOCOPS agents much safer under strong attacks, which means that our adversarial training method is generalizable to different safe RL methods.

Table 6: Evaluation results of natural performance (no attack) and under MAD, MC, and MR attackers of FOCOPS. Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds.

Env	Method	Natural		MAD		MC		MR	
		Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Car-Circle = 0:05	FOCOPS-vanilla	304.2 16.91	0.0 0.0	307.08 42.04	19.94 16.05	286.66 53.7	31.25 18.08	382.99 22.86	48.88 14.25
	FOCOPS(MC)	268.56 44.79	0.0 0.0	256.05 45.26	0.0 0.0	284.93 45.84	0.97 2.99	267.37 49.75	0.64 1.92
	FOCOPS(MR)	305.91 18.16	0.0 0.0	295.86 20.02	0.04 0.4	264.33 25.76	1.64 3.62	308.62 26.33	0.82 1.98
Car-Run = 0:05	FOCOPS-vanilla	509.47 11.7	0.0 0.0	494.74 11.75	0.95 1.32	540.23 12.56	27.0 17.61	539.85 11.85	25.1 17.29
	FOCOPS(MC)	473.47 5.89	0.0 0.0	460.79 7.76	0.0 0.0	495.54 9.83	0.45 1.15	497.24 6.6	0.62 1.23
	FOCOPS(MR)	486.98 5.53	0.0 0.0	434.96 19.79	0.0 0.0	488.24 23.98	0.62 1.1	488.58 24.65	0.52 0.9

The experiment results of trained safe RL policies under the mixture of MC and MR attackers are shown in Figure 4 and some detailed results are shown in Table 7. The mixed attacker is computed as the linear combination of MC and MR, namely, $w \quad MC + (1 \quad w) \quad MR$, where $w\ 2\ [0; 1]$ is the weight. Our agent (ADV-PPOL) with adversarial training is robust against the mixture attacker. However, there is no obvious trend to show which weight performs the best attack. In addition, we believe the performance in practice is heavily dependent on the quality of the learned reward and cost Q functions. If the reward Q function is learned to be more robust and accurate than the cost Q function, then giving larger weight to the reward Q should achieve better results, and vice versa.

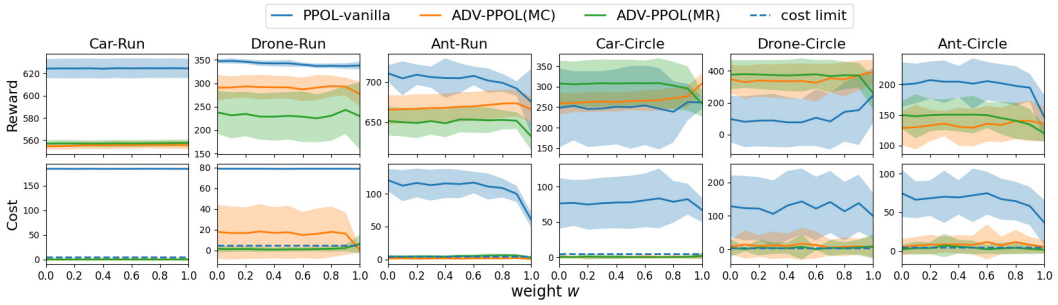


Figure 4: Reward and cost of mixture attackers of MC and MR

Table 7: Evaluation results under different ratio of MC and MR attackers of PPOL. Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds.

Env	Method	MC:MR=3:1		MC:MR=1:1		MC:MR=1:3	
		Reward	Cost	Reward	Cost	Reward	Cost
Car-Circle = 0:05	PPOL-vanilla	247.75 92.34	84.68 40.6	253.11 96.46	78.25 35.53	241.91 98.26	75.22 38.55
	ADV-PPOL(MC)	268.6 26.79	0.7 3.13	261.98 26.12	0.18 1.27	262.14 23.26	0.77 3.83
	ADV-PPOL(MR)	307.76 58.0	0.08 0.8	306.8 56.03	0.39 2.54	308.17 57.43	0.6 3.5
Car-Run = 0:05	PPOL-vanilla	624.3 8.55	183.95 0.43	624.68 8.85	184.22 0.45	624.54 8.89	184.03 0.41
	ADV-PPOL(MC)	555.45 3.37	0.03 0.18	555.64 3.47	0.02 0.13	555.22 3.09	0.0 0.0
	ADV-PPOL(MR)	557.05 3.06	0.02 0.13	557.07 3.05	0.08 0.28	556.9 2.96	0.08 0.28

The experiments for the minimizing reward attack for our method are shown in Table 8. We can see that the minimizing reward attack does not have an effect on the cost since it remains below the constraint violation threshold. Besides, we adopted one SOTA attack method (MAD) in standard RL as a baseline, and improve it (AMAD) in the safe RL setting. The results, however, demonstrate that they do not perform well. As a result, it does not necessarily mean that the attacking methods and robust training methods in standard RL settings still perform well in the safe RL setting.

Table 8: Evaluation results under Minimum Reward attacker. Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds.

Method		Car-Run ϵ = 0.05	Drone-Run ϵ = 0.025	Ant-Run ϵ = 0.025	Ant-Circle ϵ = 0.025
PPOL-vanilla	Reward	496.65±9.38	265.06±3.23	498.42±98.93	67.9±27.17
	Cost	0.0±0.0	0.0±0.0	0.03±0.18	1.17±4.52
ADV-PPOL(MC)	Reward	491.95±4.05	211.16±32.02	548.0±15.27	86.26±27.07
	Cost	0.0±0.0	0.4±1.5	0.0±0.0	0.0±0.0
ADV-PPOL(MR)	Reward	491.48±2.96	214.25±17.23	524.24±65.39	87.24±37.51
	Cost	0.0±0.0	1.1±3.14	0.0±0.0	1.4±4.59

Table 9: Evaluation results of natural performance (no attack) and under MAD, MC, and MR attackers of CVPO. Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds.

Env	Natural		MAD		MC		MR	
	Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Car-Circle $\epsilon = 0.05$	412.17 13.02	0.02 0.13	236.93 72.79	49.32 34.01	310.64 37.37	98.03 25.53	329.68 77.66	51.52 24.65
Car-Run $\epsilon = 0.05$	530.04 1.61	0.02 0.16	481.53 17.43	2.55 3.11	537.51 8.7	23.18 16.26	533.52 2.87	14.42 6.77

The experiments results of CVPO [5] is shown in Table 9. We can that the vanilla version is not robust against adversarial attackers since the cost is much larger after being attacked. Based on the conducted experiments of SAC-Lagrangian, FOCOPS, and CVPO, we can conclude that the vanilla version of them all suffer from vulnerability issues: though they are safe in noise-free environments, they are no longer safe under strong MC and MR attacks, which validate that our proposed methods and theories could be applied to a general safe RL setting.