Adversarial Behavior Exclusion for Safe Reinforcement Learning

Md Asifur Rahman, Tongtong Liu, and Sarra Alqahtani
Department of Computer Science, Wake Forest University
{rahmm21, liut18, sarra-alqahtani}@wfu.edu

Abstract
Learning by exploration makes reinforcement learning (RL) potentially attractive for many real-world applications. However, this learning process makes RL inherently too vulnerable to be used in real-world applications where safety is of utmost importance. Most prior studies consider exploration at odds with safety and thereby restrict it using either joint optimization of task and safety or imposing constraints for safe exploration. This paper migrates from the current convention to using exploration as a key to safety by learning safety as a robust behavior that completely excludes any behavioral pattern responsible for safety violations. Adversarial Behavior Exclusion for Safe RL (AdvEx-RL) learns a behavioral representation of the agent’s safety violations by approximating an optimal adversary utilizing exploration and later uses this representation to learn a separate safety policy that excludes those unsafe behaviors. In addition, AdvEx-RL ensures safety in a task-agnostic manner by acting as a safety firewall and therefore can be integrated with any RL task policy. We demonstrate the robustness of AdvEx-RL via comprehensive experiments in standard constrained Markov decision processes (CMDP) environments under 2 white-box action space perturbations as well as with changes in environment dynamics against 7 baselines. Consistently, AdvEx-RL outperforms the baselines by achieving an average safety performance of over 75% in the continuous action space with 10 times more variations in the testing environment dynamics. By using a standalone safety policy independent of conflicting objectives, AdvEx-RL also paves the way for interpretable safety behavior analysis as we show in our user study.

1 Introduction
In the last two decades, RL has greatly evolved demonstrating its potential in a wide range of applications including robotics [Kober et al., 2013], and autonomous driving [Grigorescu et al., 2020]. To be deployed in the real world, ensuring safety is a crucial factor that RL inherently lacks due to its exploratory learning. One way safety can be ensured in RL is by modifying its objective to optimize both the safety goals and the task learning [Kim et al., 2020] [Geibel, 2006]. In this approach, the RL agent must explore a significant number of safety-violating states which inherently leads to sub-optimal policies due to the conflict between the task learning and safety objectives [Thananjeyan et al., 2021]. Another way for ensuring RL safety is to enforce safe exploration by endowing explicit constraints. Manual specification of such constraints [Levine et al., 2018] though possible in environments with known dynamics, cannot be generalized to any slight changes in those environments. Besides, safety specifications based on estimating the environment dynamics during offline learning [Bastani, 2021] [Alshiekh et al., 2018] are not sufficient to assure that the RL agent will behave safely during runtime for two reasons. First, the details of the environment dynamics can’t be fully known at training time, which partially invalidates the initial assumptions about the environment modeling used to design the safety specifications. Second, RL agents are susceptible to even subtle perturbations in their observations and actions, known as adversarial examples [Chen et al., 2019] [Lee et al., 2020] which introduces novel perturbations in the environment model. Although a vulnerable policy under adversarial attacks cannot be regarded as truly safe in the physical world, there is little research studying the robustness of the safe RL methods against those attacks. In this paper, we investigate the following question: how can we ensure a robust safety for the RL agent without impairing its task learning under deliberate perturbations?

We propose the Adversarial Behavior Exclusion (AdvEx-RL) framework for safe RL. AdvEx-RL first trains an adversarial policy to interactively extract unsafe behaviors by maximizing the safety violations in the environment. Then, it learns a safety policy by maximizing its divergence from the adversarial policy. This approach is different from adversarial training which requires the agent to learn its task in a zero-sum game with the adversary. Instead, we train the adversarial policy to estimate the safety of each encountered state and to learn the underlying behavior most likely violates the safety constraints in the environment. Our contributions are three-folded: (1) developing a task-agnostic safety learning framework, AdvEx-RL, where the RL agent can use it as a safety firewall to avoid unsafe behaviors, (2) introducing

483
AdvEx-RL Safety Framework

Figure 1: AdvEx-RL Safety Framework

2 Related Work

We adopt 3 algorithmic concepts from the literature: (1) safe exploration via online shielding (2) value function-based safety estimation, and (3) the use of two policies. AdvEx-RL integrates the shielding concept from Alshiekh et al., 2018; ElSayed-Aly et al., 2021; Bansal et al., 2017; Fisac et al., 2019] to prevent the agent from visiting unsafe states during runtime. We employ a post-posed [Alshiekh et al., 2018] shield in AdvEx-RL, but instead of deriving the safety probabilities using formal methods, we use the value-function based safety estimation [Geibel and Wysotzki, 2005], [Hans et al., 2008; Srinivasan et al., 2020; Thananjeyan et al., 2021].

In [Geibel and Wysotzki, 2005; Hans et al., 2008], Q-learning is used for the risk estimation and to derive the risk-averse and rescue policies [Mihatsch and Neuneier, 2002]. The safety violations is estimated in [Srinivasan et al., 2020] through a safety critic implemented using DQN. Then, the safety Q-function estimation is utilized to optimize a Lagrangian relaxation (LR) objective to derive a safety policy. [Srinivasan et al., 2020] shows that any policy constrained under a safety Q-function is guaranteed to be safe. RecoveryRL [Thananjeyan et al., 2021] adopts the concept of safety Q-function [Srinivasan et al., 2020] to develop a shielding mechanism. They use offline demonstration data collected from human-supervised policy to train a safety estimator, Q-risk then derives a model-free recovery policy by defining an LR objective function that minimizes the Q-risk. The performance of [Thananjeyan et al., 2021] is greatly reliant on the human-supervised offline data. In AdvEx-RL we employ a similar shielding mechanism by applying a threshold on the safety estimation from a critic network [Srinivasan et al., 2020; Thananjeyan et al., 2021]. However, instead of training a different DQN safety critic through exploring a pre-training environment [Srinivasan et al., 2020] or on human supervised data [Thananjeyan et al., 2021], we acquire the safety critic from the critic network of a trained adversary.

In AdvEx-RL, we follow the strategy of using two separate policies i.e. task policy and safety policy similar to Bastani, 2021; Thananjeyan et al., 2021. In [Bastani, 2021], a shield mechanism is used to switch between the task and safety policies. The safety policy is derived using a non-linear model predictive controller (NMPC) that in the advent of probable safety violations, resets the agent to some fixed initial safety point within the environment. Whereas [Thananjeyan et al., 2021] uses a similar concept but instead of resetting the agent to some fixed initial point, they reset the agent to nearby safe points. The safety policy in NMPC [Bastani, 2021] and MPC [Thananjeyan et al., 2021] requires prior knowledge about the environment dynamics or demonstration data.

3 Problem Statement

We consider the standard CMDPs, \( \mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, \mathcal{P}, (\mathcal{R}, \gamma, C), \mathcal{C}) \) where \( \mathcal{S} \) and \( \mathcal{A} \) denote the state and action space; \( \mu \) and \( \mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1] \) denote the initial state distribution and state transition dynamics respectively. \( \mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{R} \) is the reward function; \( \gamma \) denotes discount factor; and \( \mathcal{C} = \{ c_i : \mathcal{S} \times \mathcal{A} \to \mathcal{R} \geq 0; i = 1, 2, ... T \} \) denotes the set of cost associated with safety constraint violations in any trajectory episode \( \tau = \{ s_0, a_0, ..., a_{T-1}, s_T \} \) with a maximum trajectory length of \( T \). We assume that accomplishing the task goal or violating a safety constraint in \( \mathcal{M} \) leads to episode termination. Let \( \Pi \) be the set of stationary MDP policies for \( \mathcal{M} \) such that \( \pi^{\text{task}}, \pi^{\text{ade}}, \pi^{\text{safety}} \in \Pi \) denote the task policy, adversary policy and safety policy respectively. The objective of task policy \( \pi^{\text{task}} \) is to learn the optimal control to maximize the expected discounted reward at time \( t \); \( \mathcal{R}^{\pi^{\text{task}}} = \mathbb{E}_{\tau \sim \pi^{\text{task}}} [\sum_{t=1}^{T} \gamma^{t-1} r_t] \). On the other hand, the objective of adversary policy \( \pi^{\text{ade}} \) is to maximize the expected discounted cost associated with safety violations \( \mathcal{C}^{\pi^{\text{ade}}} = \mathbb{E}_{\tau \sim \pi^{\text{ade}}} [\sum_{t=1}^{T} \gamma^{t-1} c_t] \). AdvEx-RL works like a protective safety firewall integrated with a safety shield allowing task policy \( \pi^{\text{task}} \) and safety policy \( \pi^{\text{safety}} \) to be
learned and executed completely independent of each other as depicted in Fig.1.

4 Adversarial Behavior Exclusion for Safe Reinforcement Learning

AdvEx-RL (Fig.1) uses a post-posed shielding mechanism to assess the agent’s current safety and act appropriately following either its task policy or safety policy. The task policy is any conventional RL policy optimized to learn a certain task while the safety policy is a task-agnostic policy that aims to only maximize the agent’s safety in a certain environment. When the task policy takes the agent closer to a potentially dangerous/unsafe state, the safety policy is triggered by the shield to rescue the agent to a nearby safe state.

4.1 First, Learn to be Unsafe

AdvEx-RL first extracts unsafe behavior by interactively training an optimal adversary in $\mathcal{M}$. The adversary, similar to any conventional RL, learns an optimal policy $\pi_{adv}$ to maximize the accumulated cost associated with the safety violations through the exploration-exploitation principle of RL. To improve the exploration of the adversarial policy, we use off-policy learning with entropy regularization by maximizing the following objective:

$$J(\phi) = \mathbb{E}_{(s_t,a_t) \sim \rho_{\pi_{adv}}} [c(s_t,a_t) + \alpha \mathcal{H}((\pi_{adv}(.|s_t)))$$ (1)

where $\mathcal{H}((\pi_{adv}(.|s_t)))$ is the policy entropy and $\alpha$ is the temperature parameter. Using the maximum entropy policy objective [Haarnoja et al., 2018a], the optimal state-action function $Q_{\pi_{adv}}(s,a)$ can be approximated through soft Q-function that is evaluated with Bellman backup operator $\pi_{adv}$ as:

$$\pi_{adv} Q_{\pi_{adv}}(s_t,a_t) \triangleq c(s_t) + \mathbb{E}_{s_{t+1} \sim \rho}[V(s_{t+1})]$$ (2)

where,

$$V(s_t) = \mathbb{E}_{a_t \sim \pi_{adv}}[Q_{\pi_{adv}}(s_t,a_t) - \log \pi_{adv}(a_t|s_t)]$$ (3)

The optimal $Q_{\pi_{adv}}(s,a)$ provides the estimation of the expected cost over any trajectory $\tau \in \mathcal{M}$; $Q_{\pi_{adv}}(s,a) = \mathbb{E}_{\tau \sim \pi_{adv}}[\sum_{t=0}^{T} \gamma^t c(s_t)]$. Therefore, we approximate $Q_{\pi_{adv}}(s,a)$ by minimizing soft bellman residual [Haarnoja et al., 2018a] and use it to quantify safety violation as we will show in Section 4.3. (Details on the training of the adversarial policy are given in Appendix A, Algorithm 1.)

4.2 Learn to be Safe From The Worst Behavior

The second phase of AdvEx-RL is to learn a task-agnostic safety policy exploiting the adversarial policy $\pi_{adv}$ under the following assumptions:

Assumption 1: Given CMDP $\mathcal{M}$, with the state space $S = S_{safe} \cup S_{unsafe}$, if a state $s \in S_{safe}$ has any neighboring unsafe state $\tilde{s} \in S_{unsafe}$, then the transition probability denoting safety violation upon taking an available action $a$ at that state $P(\tilde{s}|s,a) > 0$.

Assumption 2: The adversarial policy has sufficiently explored $\mathcal{M}$ and is optimal $\pi_{adv}$ such that $Q_{\pi_{adv}} \geq Q_{\pi_{adv}}$ where $\pi_{adv}$ is any sub-optimal adversary policy. Therefore $Q_{\pi_{adv}}(\cdot)$ can quantify the expected cost for any trajectory $\tau \in \mathcal{M}$.

Assumption 3: Every state has at least one neighboring safe state i.e. at any state $s_t$, there exists at least one safe action that can lead the agent to a neighboring safe state; $\exists a_t \ s.t. \ P(s_{t+1}|s_t, a_t) > 0$ where $s_{t+1} \in S_{safe}$.

Theorem 1 (Reduction of safety violation probability): The probability of following a trajectory that violates safety can be reduced by increasing KL divergence between the adversarial policy $\pi_{adv}$ and any arbitrary policy $\pi$.

Proof

Let’s consider the objective of the optimal adversary in $\mathcal{M}$ which maximizes the expected cost (Eq.1). Stationary policy $\pi_{adv} \sim \pi_{adv}$ represents a one-to-one correspondence [Puterman, 2014] with the state-action distribution in $\mathcal{M}$ which can be computed by:

$$\rho_{\pi_{adv}}(s,a) = \mu(s_0) \prod_{t=1}^{T} P(s_t|s_{t-1}, a_t) \pi_{adv}(a_t|s_{t-1})$$ (4)

From inverse reinforcement learning principle [Ghasemipour et al., 2019], the expected return of an arbitrary policy $\pi$ can be computed with respect to the optimal adversary policy (i.e. expert) $\pi_{adv}$ as:

$$\mathbb{E}_{\tau \sim \pi}[\sum_t c(s_t, a_t)] = \mathbb{E}_{\tau \sim \pi}[\sum_t \rho_{\pi_{adv}}(s_t, a_t) \frac{\rho_{\pi}(s_t, a_t)}{\rho_{\pi_{adv}}(s_t, a_t)} \log \frac{\rho_{\pi}(s_t, a_t)}{\rho_{\pi_{adv}}(s_t, a_t)}]$$ (5)

$$\propto \mathbb{E}_{(s,a) \sim \rho_{\pi}}[\rho_{\pi_{adv}}(s_t, a_t) / \rho_{\pi_{adv}}(s_t, a_t)]$$

$$= -D_{KL}(\pi || \pi_{adv})$$

This intuitively means that, as an arbitrary policy $\pi$ becomes more similar to an optimal adversary $\pi_{adv}$ by minimizing the KL divergence $D_{KL}(\pi || \pi_{adv})$, the expected cost associated with safety violations increases. Meanwhile, according to the probabilistic inference [Levine, 2018], the probability of choosing a trajectory $\tau$ involving safety violation under $\pi_{adv}$ can be given by

$$p_{\pi_{adv}}(\tau) \propto \mathbb{E}_{\tau \sim \pi}[\sum_t \gamma^t c_t]$$ (6)

Then optimal $\pi_{adv}$ can be denoted as:

$$log p_{\pi_{adv}}(\tau) = log \int p_{\pi_{adv}}(\tau) d\tau$$

$$\geq \mathbb{E}_{\tau \sim \pi}[\sum_t \gamma^t c_t] - D_{KL}(\pi || \pi_{adv}(\tau))$$ (7)
The KL divergence term in the above equation acts like a penalty regularization [Goo and Niekum, 2022] that guides the arbitrary policy $π′$ closer to the optimal adversary $π$. $D_{KL} → 0$ which in turn maximizes the expected cost. This also indicates that if the trajectory distribution under the arbitrary policy $π$ diverges from the trajectory distribution of the optimal adversary, it will then minimize the expected cost. Since diverging from a trajectory distribution is lower bounded by its state distribution [Ke et al., 2021] and considering assumption 3, maximizing KL divergence of the arbitrary policy with respect to the optimal adversary policy guarantees to reduce the probability of selecting a trajectory leading to safety violations.

We use this theorem as the learning principle of the safety policy, where the state-action distribution of policy $π_{safe}$ is derived by maximizing its KL-divergence from $π_{adv}$ according to the following objective:

$$J(θ) = \arg\max_θ E_{π_{safe}}[D_{KL}(π_{safe}(τ) || π_{adv}(τ))]$$

Eq.8 is trained by taking samples from a set of trajectories $τ_{π_{safe}}$ where the distribution of any trajectory $τ = \{s_0, a_0, s_1, a_1, ..., a_T, s_T\}$ under $π_{safe}$ is given by $ρ_{π_{safe}}(τ) = µ(s_0) \prod_{t=1}^{T} π_{safe}(a_t|s_t)P(s_{t+1}|s_t, a_t)$ then:

$$D_{KL}(π_{safe}(τ)||π_{adv}(τ)) = \sum_{τ \sim τ_{π_{safe}}} \log[\frac{π_{adv}(a_t|s_t)}{π_{safe}(a_t|s_t)}]$$

$$= \mathbb{E}_{(s_t, a_t) \sim π_{safe}}[\log π_{safe}(a_t|s_t)]$$

$$− \log π_{adv}(a_t|s_t)$$

(9)

Here the KL divergence $D_{KL}(π_{safe}(τ)||π_{adv}(τ))$ is upper bounded by Pinsker’s inequality i.e. the square root of total variational distance between the trajectory distribution under $π_{safe}$ and $π_{adv}$. To ensure that $π_{safe}$ is not only divergent from $π_{adv}$ but also will rescue the agent to a safe state, we change the objective in Eq.8 to state-pairwise safety learning by optimizing $π_{safe}$ as:

$$J(θ) = \arg\max_θ E[\sum_{τ \sim τ_{safe}} s_t \sim π_{safe}, a_t \sim π_{safe}(s_t)logρ_θ(π_{safe}(a_t|s_t))$$

$$− (Q_{φ}^{est_{adv}}(s_t, a_t) − logπ_{adv}(a_t|s_t))]$$

(10)

Where $Q_{φ}^{est_{adv}}(s_t, a_t)$ is derived by minimizing objective:

$$J(ψ) = \sum_{(s_t, a_t) \sim τ_{safe}, a_{t+1} \sim π_{safe}(a_t)} \frac{1}{2}(Q_{φ}^{est_{adv}}(s_t, a_t)$$

$$− (C(s_t) + γQ_{φ}^{adv}(s_{t+1}, a_{t+1}))^2)$$

(11)

Maximizing Eq.10 ensures that the trajectory distribution of the safety policy $π_{safe}$ is diverse from the trajectory distribution of $π_{adv}$ while increasing the probability of sampling trajectories that have lower expected safety violation cost. Consequently, $π_{safe}$ will rescue the agent to nearby states that have lower values for the adversary or in other words safer for the agent.

Sampling trajectories directly under $π_{safe}$ would lead to poor exploration. Therefore, we train $π_{safe}$ using a roll-out mechanism. This mechanism generates a demo trajectory $τ_{demo}$ using either the adversarial policy $π_{adv}$, the task policy $π_{task}$, or randomly sampled actions. For each state $s_t \in τ_{demo}$, safety policy $π_{safe}$ generates new trajectories $τ_{π_{safe}} \sim (s_t, s_{t+1}, C(s_t))$ which are then used to train the safety policy $π_{safe}$ by maximizing the objective function of Eq.10 in an off-policy fashion. (More details on AdvEx-RL safety policy training can be found in Appendix B, Algorithm 2).

4.3 Online Execution with Safety Shielding

We adopt the strategy of using a post-posed shield from [Alshiekh et al., 2018] only during the online execution. Unlike [Srinivasan et al., 2020; Thananjeyan et al., 2021], which separately trains a DQN safety critic function $Q_{risk}$ for safety estimation, we instead utilize the critic function of the optimal adversary $Q_{φ}^{safe}$ which we trained previously (Eq.2) to implement the safety shield as:

$$Shield(s_t, a_t) : Q_{φ}^{adv}(s_t, a_t) > T_{safe}$$

(12)

where $T_{safe}$ is a predefined threshold value such that at any state $s_t$ and for any action $a_t \sim π_{task}(s_t)$, if $Shield(s_t, a_t)$ is triggered, then the AdvEx-RL safety firewall replaces the selected action $a_t$ by a safer action given by the safety policy $a_t^{safe} \sim π_{safe}(s_t)$. The value of $T_{safe}$ is environment-specific and can be chosen based on a sensitivity test for each environment (see Appendix C for details about the sensitivity test.

Algorithm 3 in Appendix D shows the online execution of AdvEx-RL.)

Deadlock Side Effect: Providing safety in 2 separate policies is prone to deadlock as the agent may loop between the same states due to switching between $π_{safe}$ and $π_{task}$. Suppose at any timestep $t$, for action $a_t \sim π_{task}(s_t)$ at a critical state $s_t$, the shield is triggered. From this state $s_t$, onward, the shield will make sure that the agent keeps following a safe trajectory using $π_{safe}$ until it reaches a safe state $s_t$, such that for action $a_t \sim π_{task}(s_t)$, $Q_{φ}^{adv}(s_t, a_t) < T_{safe}$ is satisfied. Afterward, the agent can select actions using its task policy $π_{task}$ unless the shield is triggered again. However, due to the presence of external perturbation or inherent weakness within the task policy $π_{task}$, the agent might cycle back to the same old critical state $s_t$, while selecting action $a_t$: resulting in an inadvertent deadlock.

Although deadlock can hamper the agent’s task, it can save the agent from dangerous situations. For example, a deadlock can be a safe, temporary solution for an expensive robot until it gets rescued by human operators. To examine the proposed AdvEx-RL for deadlocks, we empirically analyze the frequency of deadlocks in the tested environments using the deadlock detection proposed in [Ye et al., 2022].

5 Practical Implementation

The adversarial policy was trained using SAC [Haarnoja et al., 2018b] since it provides better exploration. The task policy was also trained using SAC but it can be trained using any
RL algorithm. The safety policy was trained by performing gradient descent on the objective function in Eq.10. The postposed shield was implemented as a safety assurance layer that explicitly replaces any unsafe action selected by the task policy with a safe action chosen by the safety policy during execution.

6 Experiments

The experiments in this paper are conducted to answer the following questions: (1) how robust is AdvEx-RL compared to the baselines under deliberate uncertainty in form of external perturbations and altered environment dynamics? (2) how does AdvEx-RL’s safety policy affect the agent’s task performance? (3) how much does the safety policy contribute to the safety of AdvEx-RL? (ablation analysis) (4) how often does the deadlock occur in AdvEx-RL and the baselines? and (5) how transparent and interpretable is the behavior generated by AdvEx-RL to the end users? All the codes relevant to the experiments are available online.

6.1 Environments

We conducted our experiments on three continuous MuJoCo CMDPs [Thananjeyan et al., 2021] (i) Maze (ii) Navigation 1, and (iii) Navigation 2. In these environments, the agent’s task is to reach the goal state while avoiding collisions with obstacles, walls, or boundaries. In addition, we also conducted experiments on SafetyGym environments [Ray et al., 2019]. (See Appendix E for more details about the environments.)

6.2 Baselines

We have tested AdvEx-RL against 7 baselines; SAC (without any safety measures), Lagrangian Relaxation (LR) [Thananjeyan et al., 2020], Safety Q-Functions for RL (SQRL) [Srinivasan et al., 2020], Risk Sensitive Policy Optimization (RSPO) [Mihatsch and Neuneier, 2002], Critic Penalty Reward Constrained Policy Optimization (RCPO) [Tessler et al., 2018], Reward Penalty (RP) [Thananjeyan et al., 2021], and Recovery RL Model Free (RRL-MF) [Thananjeyan et al., 2021]. (More details on the baselines are in Appendix F. In addition, see Appendix G for further implementation details of AdvEx-RL and the baselines.)

6.3 Performance Metrics

Assuming a maximum episode length $T_{max}$ in any environment, the following cases might happen: (1) the agent accomplishes its task within $T_{max}$ without any safety violations, (2) it violates at least one safety constraint and terminates, or (3) it exhausts $T_{max}$ without accomplishing its task or violating any safety constraints. Considering these cases, we use the following two performance metrics in our experiments: (i) Safety(%): This metric measures the portion of time the agent acts safely over its maximum episode length. Given $T_{max}$, the function $F(.)$ counts the total number of timesteps before the episode termination caused by safety violation. Then Safety% is measured as a function of trajectory $\tau$:

$$Safety(\%) = \begin{cases} \frac{F(\tau)}{T_{max}} \times 100 & \text{if } \exists s_t \sim \tau \text{ e.g. } C(s_t) > 0 \\ 1 \times 100 & \text{otherwise} \end{cases}$$

(ii) Success-Safety(%): Assuming the AdvEx-RL agent’s task is to reach a goal state $G$, both the reward and success are measured in terms of how close the agent is to $G$. The agent is considered successful in accomplishing its task when the episode ends while it is within a predefined minimum distance from $G$. The maximum distance from $G$ is given by $Max_{distance}$ and the agent’s current Euclidean distance from the goal is $D(s_t, G)$, then the Success-Safety % measures the trade-off between safety and task objectives by:

$$Success - safety(\%) = \frac{D(s_t, G) - Max_{distance}}{Min_{distance} - Max_{distance}} \times Safety(\%)$$

6.4 Robustness Analysis

While prior safe RL works focus on the performance optimality, we argue that optimality is not enough and we have to study the robustness of the safety solutions against deliberately crafted perturbations. Therefore, we evaluate the robustness of the safety performance of the baselines and AdvEx-RL under different magnitudes and types of uncertainty. We injected two types of uncertainty into the testing environments: external action space perturbations and changes to the environment dynamics (such as air resistance and noise). For the first type, we kept the environment dynamics the same as the training environment while incorporating two white-box action space perturbations; random and alternative adversarial action (AAA) perturbations from [Tessler et al., 2019]. (See Appendix H for details.)

6.5 Results

Evaluation of Safety Robustness: The results of the safety robustness analysis are presented in Fig.2 and Fig.3. The results are averaged over 100 test episodes. Evidently, the baselines showed optimal safety performance under no external perturbations or dynamics changes. However, a significant deterioration of both safety(%) and success-safety(%) can be seen for most of the baselines with increasing in perturbation rates. Performance deterioration is more visible in the case of the AAA perturbation than random perturbation, which indicates that the prior safe RL techniques are not robust against deliberate perturbations. Moreover, changes in the environment dynamics negatively impact the safety robustness of the baselines as well as AdvEx-RL. Interestingly, in Fig.2, we observe that SAC provides reasonable safety which is due to its default robustness properties [Eysenbach and Levine, 2021]. RRL-MF provides the second-best safety robustness, particularly for higher perturbation rates and the reason for this could be its usage of explicit unsafe demonstration data provided by experts during its training. In both uncertainty types, AdvEx-RL clearly outperforms the baselines with a minimum safety(%) over 75% and success-safety(%) over 80%. This empirically proves the robustness of AdvEx-RL.

3https://github.com/asifurrahman1/AdvEx-RL
Figure 2: Safety(%) and success-safety(%) performance of the baselines and AdvEx-RL under the influence of various rates of external action perturbations in the Maze (top row) and Navigation 2 (bottom row) environments suggest that baseline methods are not robust, and their performance decreases when the attack rate increases. During this analysis, the dynamics of the environments were kept the same during training and testing. (See Appendix H for results on Navigation 1).

Figure 3: The robustness performance of AdvEx-RL and baselines on the Navigation 1 test environment with 10 times more variation in dynamics than its training environment while being exposed to external perturbations. The results show that AdvEx-RL is more robust to variations in the environment dynamics than other baselines. (Please refer to Appendix K for the detailed experiment).

Random Action Perturbation

AAA Perturbation

Ablation analysis: AdvEx-RL, has three components that contribute to its safety performance i.e (1) the task policy, (2) the safety policy, and (3) the shield. To evaluate the contribution of the safety policy to the overall performance, we conducted a thorough ablation study, the details of which can be found in Appendix L.

Deadlock analysis: We analyzed the presence of deadlock in the baselines while subjected to different rates of AAA perturbation (ranging from 0% to 100%). Using a 20-step look-ahead for deadlock cycle detection across 100,000 test episodes, we found no deadlock cycles in the baselines and AdvEx-RL in all three environments. (See code supplements for deadlock experiment details).

Interpretability Analysis: Since safety-assurance approaches, in general, compromise system performance, we must ensure that human practitioners and users trust them, lest they ignore them and negate their effectiveness. Trustworthiness regarding how safe an agent is depends on how transparent its behavior is to the end users. For the joint task-safety optimization techniques [Srinivasan et al., 2020;...]

488
Tessler et al., 2018; Liu et al., 2022), it is not possible to pinpoint what influences the agent’s behavior, task, or safety objective. AdvEx-RL, however, uses two separate policies for the task and safety and, therefore, capable of explaining what influences its behavior at each time step. To analyze the interpretability of AdvEx-RL, we extend the explainable RL method CAPS [McCalmon et al., 2022] into safety-CAPS to explain the impact of safety violations on the agent’s behavior and to provide a directed graph with an explanation of the agent’s safety policy by extending CAPS with two criteria: risk estimation and the episode length in time steps. We use the safety-CAPS graphs to answer the question of “why the agent is taking certain actions at certain states?” We then add more details to the CAPS graph about how, when, and why the safety policy and safety shield are triggered during safety violations. Episode length (TS) measures the time steps the agent takes to accomplish its task. Perturbing the agent’s policy during safety violations could result in the agent taking a longer time to reach its goal state. Hence, we use this metric to answer the question of “why the agent is not taking action \( a’ \) instead of \( a \) at state \( s_c \)” by attaching the timesteps to each abstract state of the CAPS graphs. The risk estimation (RE), on the other hand, measures how likely the agent will fail its task when it takes action \( a’ \) instead of following its policy and taking \( a \) at a critical state \( s_c \) due to a safety violation. To estimate the risk, we build on the risk estimation approach proposed in [Uesato et al., 2018] for uncovering failures in RL agents by sampling the agent in states where its safety is violated. We use a neural network with two fully-connected layers of 64 neurons each, and train it to predict RE from each state \( s_t \). An example of the safety-CAPS graph for Navigation 2 is displayed in Fig.4.

![Figure 4: An example of the graphs generated by safety-CAPS for Navigation 2.](image)

We conducted a user study using the Navigation 2 environment and presented 41 Amazon Mechanical Turk (AMT) workers with the safety-CAPS graphs and asked them 8 questions (See Appendix M for the study and the details of the questions) about their interpretability of AdvEx-RL policies. A summary of the accuracy of the users’ answers to each question is shown in Figure 5.

In questions 1, 2, 5, we asked the users to identify optimal actions the agent will take at a certain state with its task policy (Q1), task policy under attack (Q2), and task policy under attack but with AdvEx-RL’s safety policy (Q3). The participants demonstrated a good understanding of the environment, with an accuracy rate above 80% in those questions. Notably, the accuracy rate of Q5 is above 90%, which indicates that the users can understand the purpose and impact of the safety policy on the agent’s behavior. This clearly shows the interpretability of AdvEx-RL for end-users regardless of their RL background.

Q3 and Q6 are true or false questions where we asked the users to identify if the agent will terminate in a dangerous state under an attack without (Q3) or with (Q6) the safety policy. Roughly 70% and 85% of the users correctly answered Q3 and Q6, respectively. The increase in the accuracy of Q6 demonstrates that the safety-CAPS graph with the safety policy can better convey the reason behind the agent’s actions. Q4 and Q7 are counterfactual reasoning type questions where we asked the users why the agent terminated in dangerous states under attack (Q4) but successfully avoided the dangerous states using the safety policy (Q7). We think the low accuracy for those questions is due to their difficulty and demand in terms of logic and analytical skills, which can be challenging to non-technical participants who do not have RL background. Lastly, Q8 is a comprehensive measure of the users’ understanding of the overall impact of the safety policy on the agent’s behavior. Approximately 73% of the users successfully understood how the safety policy protects the agent from choosing unsafe actions. At the same time, we also need to consider the fact that the length of the study and the dependent relationship of questions generally pose challenges to the participants. If they fail to understand the central idea of the study, they tend to perform badly subsequently. All the participants who incorrectly answered Q8 have at least answered 2 or 3 questions incorrectly before. Therefore, we believe our AdvEx-RL is interpretable by non-technical users using the graphs generated by safety-CAPS.

![Figure 5: The summary of the accuracy of users’ answers to the 8 questions in the user study.](image)

### 7 Conclusion

In this paper, we introduced an alternative view on safety learning for RL through our task-agnostic safety framework AdvEx-RL. We empirically showed that AdvEx-RL is effective in ensuring safety even in uncertain conditions. Through a user study conducted on 41 non-technical end users, we also demonstrated the transparency of AdvEx-RL by explaining its behavior using safety-CAPS. In future work, we plan to extend this framework to multi-agent settings along with an explainable safety method.
Acknowledgments
This material is based upon work supported by the National Science Foundation (NSF) under grant no. 2105007.

References
[McCalmon et al., 2022] Joe McCalmon, Thai Le, Sarra Alqahtani, and Dongwon Lee. Caps: Comprehensible abstract policy summaries for explaining reinforce-


