xSRL: Safety-Aware Explainable Reinforcement Learning - Safety as a Product of Explainability

Risal Shahriar Shefin
Wake Forest University
Winston-Salem, NC, US
shefrs24@wfu.edu

Md Asifur Rahman
Wake Forest University
Winston-Salem, NC, US
rahmm21@wfu.edu

Thai Le
Indiana University
Bloomington, IN, US
tle@iu.edu

Sarra Alqahtani
Wake Forest University
Winston-Salem, NC, US
sarra-alqahtani@wfu.edu

ABSTRACT

Reinforcement learning (RL) has shown great promise in simulated environments, such as games, where failures have minimal consequences. However, the deployment of RL agents in real-world systems such as autonomous vehicles, robotics, UAVs, and medical devices demands a higher level of safety and transparency, particularly when facing adversarial threats. Safe RL algorithms aim to address these concerns by optimizing both task performance and safety constraints. However, errors are inevitable, and when they occur, it is essential that RL agents can explain their actions to human operators. This makes trust in the safety mechanisms of RL systems crucial for effective deployment. Explainability plays a key role in building this trust by providing clear, actionable insights into the agent's decision-making process, ensuring that safety-critical decisions are well understood. While machine learning (ML) has seen significant advances in interpretability and visualization, explainability methods for RL remain limited. Current tools fail to address the dynamic, sequential nature of RL and its need to balance task performance with safety constraints over time. The re-purposing of traditional ML methods, such as saliency maps, is inadequate for safety-critical RL applications where mistakes can result in severe consequences. To bridge this gap, we propose xSRL, a framework that integrates both local and global explanations to provide a comprehensive understanding of RL agents' behavior. In addition, xSRL enables developers to identify policy vulnerabilities through adversarial attacks, offering tools to debug and patch agents without retraining. Thus, xSRL enhances the RL safety as a byproduct of explainability and transparency. Our experiments and user studies demonstrate xSRL's effectiveness in increasing safety in RL systems, making them more reliable and trustworthy for real-world deployment. Code is available at https://github.com/risal-shefin/xSRL

KEYWORDS

safety, explainability, RL, global explanation, vulnerabilities

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

ACM Reference Format:
Risal Shahriar Shefin, Md Asifur Rahman, Thai Le, and Sarra Alqahtani. 2025. xSRL: Safety-Aware Explainable Reinforcement Learning - Safety as a Product of Explainability. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) has demonstrated significant potential in simulated environments, such as games and simplified settings, where failures have minimal consequences. However, deploying RL agents in real-world systems—such as autonomous vehicles, robotics, UAVs, and medical devices—introduces a much higher risk, as failures can lead to severe repercussions. Thus, training RL agents in these settings requires a greater emphasis on safety, especially under adversarial conditions. Addressing these concerns, the field of safe RL has emerged, developing algorithms that explicitly optimize both task performance and safety constraints [1, 5, 7, 16, 18, 22, 31, 33]. While these algorithms aim to enhance both safety and accuracy, errors are inevitable, and when they occur, it becomes crucial for the agent to explain its behavior to human practitioners. Practitioners must discern whether an agent is behaving correctly, or is malfunctioning and thus requires intervention. Therefore, user trust in these safety approaches is vital; without it, practitioners may disregard the systems, undermining their effectiveness. Explainability fosters this trust by allowing practitioners to query the reasoning behind safety-related decisions, alongside the agent's learned policy. This additional transparency helps make safety decisions more actionable and strengthens practitioners' confidence in the RL system. In this work, we hypothesize that *safety in RL is intrinsically tied to explainability and transparency*, which allow the users to gain a clear understanding of agent behavior, enable efficient debugging and address its safety concerns.

The need for explainable RL (XRL). In machine learning (ML), significant progress has been made in developing interpretation and visualization methods to help users understand a ML model's performance, track its metrics [27], generate data-flow graphs for its decision-making [39], and visualize representations it has learned [36]. However, interpretation tools designed specifically for RL agents are still limited, most of which only offer basic capabilities such as behavior summarization [3], contrastive explanations [37], or short video clips at critical time steps, but without providing

any in-depth reasoning about the RL agents' behavior. While repurposing ML interpretation methods (e.g., saliency maps [30]) might seem feasible, their design intent is insufficient for explaining RL agents, especially in safety-critical environments. RL agents must not only optimize task performance but also adhere to strict safety constraints, as any mistake can lead to significant harm to agents or human workers. Existing ML explainability tools focus on individual, static decisions, which fails to capture the sequential and dynamic nature of RL, where actions impact future states and they require balancing immediate rewards with long-term risks. Additionally, RL agents often operate under stochastic policies and delayed feedback, making it difficult to explain how they learn to manage the trade-off between exploration and exploitation while maintaining safety. Therefore, safety-aware explainability must address both task performance and the agent's adherence to safety constraints, ensuring that every action is evaluated in terms of both its short- and long-term benefits and risks—a complexity that re-purposed ML tools are not well-equipped to handle.

Desiderata. Effective XRL methods for safety must meet key criteria to ensure usability and trustworthiness. First, they must provide global explanations of the agent's policy, offering insights into its behavior across the state space [10, 21, 35, 40], especially for end-users unfamiliar with RL. Second, they should deliver local explanations, detailing the agent's reasoning at specific timesteps. Third, explanations must be interpretable, preferably in natural language, with minimal user effort. Fourth, the method must ensure explanation fidelity, validated through tests [20] to confirm alignment with the agent's actual policy. Finally, XRL should include adversarial explainability, analyzing how decisions change under adversarial conditions to identify vulnerabilities and improve robustness. Beyond user trust, XRL should provide utilities for RL users and developers, such as debugging, testing, and patching policies to enhance safety and performance [6, 12, 24]. Currently, no tools or frameworks address all these needs.

Proposal. To address the desiderata, we propose xSRL, a novel framework integrating local and global explanations to enhance RL safety through transparency and explainability. xSRL explains safety violations by showing how task requirements or safety constraints influence decisions. xSRL introduces a novel local explanation method, training two critics, Q_{task} and Q_{risk}, to estimate returns and risks for a state s_t, action a_t, and policy π. These post-hoc Q-functions apply to any fixed policy without accessing its internal structure. For global explanations, xSRL extends our previous work, CAPS [21], enhancing nodes (abstract states) with Q_{task} and Q_{risk}, and annotating edges (actions) with motivations rooted in task or safety requirements. xSRL also provides adversarial explainability, enabling developers to debug and patch policies by identifying vulnerabilities and analyzing behavior under risk. Developers can launch adversarial attacks and improve safety without retraining the policy. To our knowledge, xSRL is the first framework combining local and global explanations to address RL agent safety and the first to offer adversarial explanations for vulnerability analysis and policy patching.

Key Contributions. We show that xSRL' explanations provide the users with both *trust* and *utility* to understand and protect RL agents in critical scenarios. Trust is measured computationally through fidelity tests and empirically via user studies. Fidelity tests assess the alignment between the explanation graphs and the agent's actual policy across two benchmark environments in safe RL [26, 33], using safety techniques [26, 31, 33] to patch vulnerable policies. Results indicate that xSRL generates accurate policy graphs with less than 33.5% error in risk estimation, even under adversarial attacks. To further validate xSRL's effectiveness, we compared xSRL against local and global explanations by evaluating users' comprehension of xSRL explanations for both vulnerable and safe policies under adversarial attacks in a user study with 9 distinct conditions. These conditions represented various combinations of local and global explanations with different safe RL patching methods. Both xSRL and global explanations (CAPS [21]) achieved comparable accuracy, demonstrating the superiority of global explanations (xSRL and CAPS) over local explanations in providing clearer insights in high-risk contexts. However, xSRL offers a distinct advantage over CAPS by facilitating debugging and patching for safety constraints. We also evaluated the impact of explanation-guided adversarial attacks on an agent trained with the soft-actor-critic (SAC) algorithm [9], where the attack reduced the agent's safety by approximately 72% under a 50% attack rate. After identifying vulnerabilities in the SAC agent's policy, we patched it with a safety shield [2] and a safe policy, significantly improving the agent's safety. Lastly, user studies assessed participants' ability to distinguish the safer agent between two alternatives, confirming xSRL's utility in enhancing safety and providing actionable insights. Overall, xSRL effectively increases the trustworthines and utility of RL agents for real-world, safety-critical applications by identifying and resolving policy vulnerabilities.

2 RELATED WORK

XRL methods have evolved to offer insights into the behavior of RL agents, with approaches generally categorized into local and global explanations [25]. *Local explanations* focus on specific actions taken by an agent at a point in time, often using post-hoc methods like saliency maps [8, 11, 14], which highlight key features that influence the decisions. Intrinsic methods, such as reward decomposition [15], build explainability into the agent's decision model itself by breaking down the Q-value into components that reveal the agent's motivations at each time step. These approaches help in understanding why specific decisions are made but do not provide insight into the satisfaction or violation of safety constraints explicitly.

Global explanations attempt to describe an agent's overall strategy by summarizing its behavior across various states. Recent work introduced the concept of "agent strategy summarization" [3, 4], where an agent's behavior is demonstrated through its actions in a carefully chosen set of states. The key challenge in this approach is how to select the most crucial state-action pairs that effectively portray the agent's behavior, allowing users to anticipate how it may act in new scenarios. Techniques such as HIGHLIGHTS [3] identify important states based on the impact of decisions on the agent's utility, while other methods use machine learning to optimize state selection [13, 17]. These summaries reduce the effort required for humans to understand the agent's behavior while still providing comprehensive information about its capabilities.

Combining local and global methods, such as integrating strategy summaries with saliency maps [14], has shown promise but lacks the depth needed for understanding safety-oriented decisions such as the estimation of the risk of agent's failing at each state and for the overall policy. In general, none of the existing XRL methods explicitly explain how an agent adheres to or violates safety constraints. Moreover, current XRL approaches primarily focus on increasing end-user trust by providing interpretable models that emphasize behavior, without evaluating how these explanations could benefit developers tasked with debugging or refining the RL agents.

Our work addresses these gaps by developing xSRL, a framework that integrates local and global explanations specifically designed to convey the satisfaction or violation of safety constraints explicitly. xSRL combines strategy summarization with safety-focused local explanations, illustrating how agents balance task performance with risk assessments across different states. We hypothesize that *"safety is a product of explainability"* by offering actionable insights that enable developers to interactively debug and refine RL policies, thereby enhancing safety. This utility is crucial for improving RL policies in safety-critical environments, where errors can have severe consequences. *To our knowledge, no existing XRL method provides such a comprehensive, safety-oriented approach, highlighting the importance and novelty of our approach.*

3 BACKGROUND

For the purpose of explaining safety of RL agents, we consider the standard Constrained Markov Decision Processes (CMDPs), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, \mathcal{P}(.|.,.), \mathcal{R}, \gamma, C)$ where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space; μ and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ denote the initial state distribution and state transition dynamics, respectively. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the reward function; γ denotes discount factor; and $C = \{c_i : \mathcal{S} \times \mathcal{A} \xrightarrow{\mathcal{S}} \mathbb{R} \geq 0;\ i = 1, 2, .., T\}$ 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 either accomplishing the task goal or violating a safety constraint in $\mathcal{M}$ leads to episode termination. The objective of the agent's policy, we call it *task policy* hereafter, π_{task} is to learn the optimal control to maximize the expected discounted reward at time t; $\mathcal{R}_{\pi_{task}} = \mathbb{E}_{\tau \sim \pi_{task}}[\sum_{t'=t}^{T} \gamma^{t'-t} r_t]$. The task *policy* π_{task} is the solution to the CMDP.

Problem Statement. Our ultimate goal is to provide a comprehensive explanation of an RL agent's behavior. Consider an RL task solved by an agent trained with either value-based algorithms such as DQN [23] and DDQN [38] or policy-based algorithms such as PPO [29] or TRPO [28]. This paper aims to explain this agent's policy by summarizing its overall strategy to solve the task. Formally, given N episodes $\mathbb{T} = \{\mathbf{X}^{(i)}, r_i, c_i\}_{i=1:N}$ of the target agent, $\mathbf{X}^{(i)} = \{s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, c_t^{(i)}\}_{t=1:T}$ is the i-th episode of length T, where $s_t^{(i)}$ is the state, $a_t^{(i)}$ is the action, $r_t^{(i)}$ is the reward, and $c_t^{(i)}$ is the cost, at time t in episode i. Our goal is to generate a summary of these episodes as a *graph* accompanied by estimated values of Q-functions for the *task reward* and the *cost of safety violations*.

Baseline. In this paper, we build on our previous method, CAPS [21], a recently introduced global explanation XRL method that has been successful in providing comprehensible Summaries of RL policies, as our main comparison baseline. CAPS collects natural language (NL) predicates from the user and gathers up to 500 timesteps from the RL agent's trajectories. To simplify the explanation process, it uses the CLTree [19] clustering algorithm to abstract the agent's states into clusters, forming a hierarchy. A heuristic optimization then selects the best cluster configuration based on state transition accuracy and user interpretability. CAPS constructs the agent's policy (π) as a directed graph $G = (V, E)$, where set of nodes V represent abstract states (clusters), and set of edges E show the agent's actions and transition probabilities. CAPS enriches the graph by labeling abstract states with English explanations based on user-defined predicates and boolean algebra. We chose CAPS [21] because it offers a comprehensive global explanation, revealing the agent's policy across the state space rather than focusing on specific states. It also outperforms other global XRL methods [35, 40] in fidelity and user comprehension.

While CAPS [21] can explain an RL agent's policy, it cannot address the impact of safety violations on the agent's behavior, nor does it enable users to debug specific safety concerns. *xSRL addresses these limitations by providing safety through enhanced explainability and transparency.*

4 APPROACH: SAFETY-AWARE EXPLAINABLE RL METHOD

In this section, we present our approach, so-called xSRL, which uniquely combines a novel local explanation method with an extended version of the global explanation framework CAPS [21] to explain an RL agent's decision-making process at both individual states and its overall strategy to achieve a comprehensive consideration for both task and safety requirements.

4.1 Safety Interpretation via Integrating Local and Global Explanations

Local Explanation Method. In XRL, reward decomposition [15] is used to reveal the reasoning behind an agent's actions in specific states by decomposing the reward into different components. This method, which separates rewards into individual reward components values $R_c(s, a)$, highlights the factors influencing an agent's decisions at each timestep. However, such approach is not built to explain safety for RL agents, which can be optimized in a separate objective function through joint optimization [22, 31, 34] or provided through a separate policy from the task policy [26, 33]. To bridge this gap, we propose a local explanation method that can explicitly explain safety in RL. Our local explanation method consists of 2 components: task reward estimation and risk estimation at each (s, a) pair.

To estimate the future risk probability of a safety-constrained task policy π_{task}, we train a separate risk-critic Q_{risk} that evaluates the safety of a given state-action pair independently of task objectives. This risk-critic function $Q_{\text{risk}}^{\pi_{\text{task}}}(s, a)$, learned via:

$$Q_{\text{risk}}^{\pi_{\text{task}}}(s, a) = \mathbb{E}_{a_t \sim \pi_{\text{task}}(.|s_t)}[\sum_{t'=t}^{T} \gamma_{risk}^{t'-t} c(s_t, a_t)], \tag{1}$$

provides a way to assess the safety constraints of actions taken by the task policy. Here, $c(s_t, a_t)$ denotes the cost associated with violating safety constraints when taking action a_t in state s_t. In practice, we approximate $\hat{Q}^{\pi_{\text{task}}}_{\phi,\text{risk}}$, parameterized by ϕ, using sampled transitions (s_t, a_t, s_{t+1}, c_t). This is done by minimizing the following MSE loss with respect to the target (RHS of Eq. 1):

$$J_{\text{risk}}(s_t, a_t, s_{t+1}; \phi) = \frac{1}{2}\Big(\hat{Q}^{\pi_{\text{task}}}_{\phi,\text{risk}}(s_t, a_t) - \mathbb{E}_{a_{t+1}\sim\pi(.|s_{t+1})}[\hat{Q}^{\pi_{\text{task}}}_{\phi,\text{risk}}(s_{t+1}, a_{t+1})]\Big)^2 \tag{2}$$

To estimate the future task reward for a task policy π_{task} at each state, we train a separate task-critic Q_{task} similar to Q_{risk} in Eq.1 but for the task reward r instead of safety cost c:

$$Q^{\pi_{\text{task}}}_{\text{task}}(s, a) = \mathbb{E}_{a_t\sim\pi_{\text{task}}(.|s_t)}[\sum_{t'=t}^{T} \gamma^{t'-t}_{\text{task}} r(s_t, a_t)] \tag{3}$$

Since value-based RL algorithms, in general, and actor-critic policy-based algorithms estimate Q-function for the task policy, we can directly utilize their Q_{task} to explain their performance at each state in terms of task requirements.

Global Explanation Method. To enhance the effectiveness of local explanations, we propose integrating both Q_{task} and Q_{risk} into the global explanation method, CAPS [21]. This approach allows users to understand how the agent balances task objectives with safety constraints while presenting its overall strategy across episodes. CAPS summarizes the agent's policy in a directed graph where nodes represent abstract states and edges represent actions along with their transition probabilities, as described in Section 3. We choose to present xSRL explanations in a directed graph format because it effectively illustrates the relationships and dependencies among states, actions, and outcomes; their causal and safety relationships; and the progression from one state to another based on specific actions. However, using CAPS alone as a global explanation tool did not provide sufficient insight for users to determine why the agent chose a particular action at a specific state, as demonstrated in our user studies (Section 5).

Our method xSRL improves CAPS graph by: (1) incorporating local Q_{task} and Q_{risk} to show task and risk estimation at each abstract state, and (2) explicitly indicating whether each action is driven by task or safety considerations. We hypothesize that combining global policy summaries with Q-function decomposition will significantly enhance user understanding of agent safety, compared to relying solely on local or global explanations, as shown in our evaluation section. To compute $Q_{\text{risk}}(B, a)$ and $Q_{\text{task}}(B, a)$ for an abstract state B and action a, we average $Q_{\text{risk}}(s, a)$ and $Q_{\text{task}}(s, a)$ over all concrete states $s \in B$ and all possible actions a from s:

$$Q^{\pi_{\text{task}}}_{\text{task}}(B, a) = \frac{1}{n}\mathbb{E}_{a\sim\pi(.|s)}\sum_{i=1}^{n} Q^{\pi_{\text{task}}}_{\text{task}}(s, a) \tag{4}$$

$$Q^{\pi_{\text{task}}}_{\text{risk}}(B, a) = \frac{1}{n}\mathbb{E}_{a\sim\pi(.|s)}\sum_{i=1}^{n} Q^{\pi_{\text{task}}}_{\text{risk}}(s, a), \tag{5}$$

where n represents the total number of concrete states within the abstract state B. These values are attached to the abstract state B in the directed graph, showing how the agent's policy π evaluates task satisfaction (Eq.4) and safety constraint violations (Eq.5) when taking action a from state B. The second enhancement we added to CAPS will be discussed in Section 4.3.

4.2 Safety Debugging via Adversarial Explanation

This section illustrates how xSRL can be used to provide so-called *adversarial explanations* to help users in discovering and debugging vulnerabilities in RL policies. Specifically, we demonstrate how users, using the information revealed by xSRL, can launch adversarial attacks to explore potential pitfalls of an RL agent. Through xSRL's graphs, users can also explain the agent's mistakes and its violations of safety constraints, allowing them to formulate a remediation policy that improves the agent's original behavior (discussed in Section 4.3).

To initiate an adversarial attack, we first collect 500 episodes (τ) from the target agent and explain them using xSRL's graph $\mathbb{G}$. Next, using $\mathbb{G}$, we identify the top-K safety-critical states across all episodes, defined as follows:

Definition 1: Safety-Critical State. *Given a policy π_{task}, a state s_c is a safety-critical state iff there is at least one action a chosen by π_{task} such that:*

$$Q^{\pi_{\text{task}}}_{\text{risk}}(s_c, a) > \epsilon_{\text{safety}}, \tag{6}$$

where *the set of all safety-critical states $\mathbb{C}$ is $\forall s_c \in S$*. Finally, we run the agent for another 500 episodes, forcing it to take adversarial actions at the common critical states $\mathbb{C}$ identified by $\mathbb{G}$ at varying rates (as we show in Section 5), while collecting its trajectories τ_A. These adversarial actions are generated using the Alternative Adversarial Action (AAA) attack [32], employing a pre-trained adversarial policy π^{adv} from [26] to select alternative adversarial actions $a_{\text{adv}} \sim \pi^{\text{adv}}(s_t)$. We then use xSRL to generate explanation graph for the agent under attack $\mathbb{G}_{\mathbb{A}}$, based on τ_A. By contrasting both graphs, $\mathbb{G}$ and $\mathbb{G}_{\mathbb{A}}$ without and with the adversarial attacks, respectively, users can pinpoint the safety vulnerabilities in the agent's policy by noticing the overall graphs and their attached Q_{risk} and Q_{task} values.

4.3 Patching Explanation-Based Discovered Vulnerabilities

This section demonstrates how xSRL can guide the patching process for vulnerabilities discovered in RL policies through adversarial explanations. To avoid retraining the task policy π_{task}, we propose a simple approach using an auxiliary policy that optimizes only the safety violation cost function c. Specifically, we employ the safety policy π_{safety} from [26], which is trained by maximizing the KL-divergence from an adversarial policy that increases safety violation costs. We then adopt the post-posed shielding strategy from [2] during online execution to shield the states with higher Q_{risk} from attacks. To implement the safety shield, we leverage the risk-critic Q_{risk} trained for explainability in (Eq.1) as:

$$Shield(s_t, a_t) : Q_{\text{risk}}(s_t, a_t) > \mathbb{T}_{\text{safety}}, \tag{7}$$

where $\mathbb{T}_{\text{safety}}$ is a predefined threshold value such that at any state s_t and for any action $a_t \sim \pi^{\text{task}}(s_t)$; if $Shield(s_t, a_t)$ is triggered, then the shield replaces the selected action a_t by a safer action given by the safety policy $a^{\text{safe}}_t \sim \pi^{\text{safety}}(s_t)$. The value of $\mathbb{T}_{\text{safety}}$

is environment-specific and can be chosen based on a sensitivity test for each environment.

xSRL can then be used to generate an updated graph for the patched agent, enabling developers to assess improvements or uncover new vulnerabilities. Since the proposed patching process, along with certain safe RL algorithms like [26, 33], involves two distinct policies—task and safety, it provides additional data to further refine the xSRL graphs. Using the shielding threshold, each edge (B, a) in the graph is labeled as a "safety decision" if action a is chosen by π_{safety}, or a "task decision" if selected by π_{task}. The responsible policy is identified by tracking which policy (safety or task) selects the actions in each concrete state $s \in B$, with the dominant policy making the most decisions in B being assigned.

5 EVALUATION

In this section, we evaluate xSRL based on two key objectives: trust and utility. Trust assesses whether the explanations generated by xSRL are comprehensible to end-users, while utility measures the effectiveness of these explanations in identifying and resolving vulnerabilities in the agent's policy.

Tasks: We conducted our experiments and user studies in two continuous MuJoCo CMDP environments [33]: (1) Navigation 2, and (2) Maze. In both environments, the state-actions spaces are continuous and the agent's objective is to reach a goal state while avoiding collisions with obstacles, walls, or boundaries. For each task, we used a well-trained SAC agent [9] as the target. We present our results for Navigation 2 in the main paper, with *the results for Maze provided in Appendix D.*

Explanation Baselines: As discussed in Section 4.1, XRL methods can be classified into two broad categories: (1) local explanations and (2) global explanations. For comparison, we select two representative baselines. The first baseline is our local explanation method, which presents users with Q_{task} and Q_{safety} values at specific time steps. The second baseline is CAPS[21], which generates a directed graph summarizing the agent's overall policy. Figure.1 illustrates three different explanations in Navigation2 using our local explanation method, global explanation (CAPS), and xSRL, which integrates both local and global explanations, for an agent patched with the safe policy from [26].

Safety Patching Baselines: We employed three different safe RL methods as patching techniques. First, we used the safety policy π_{safe} developed in AdvExRL [26] and RRL-MF [33], separately. These methods represent approaches that optimize safety separately from task performance. Second, we tested SQRL [31], which exemplifies joint optimization of both task performance and safety. Details of these methods are given in Appendix A.

5.1 Trustworthiness of xSRL's Explanations

We evaluate the trustworthiness of xSRL explanations using two approaches: computational fidelity scores and user studies.

5.1.1 *Fidelity.*

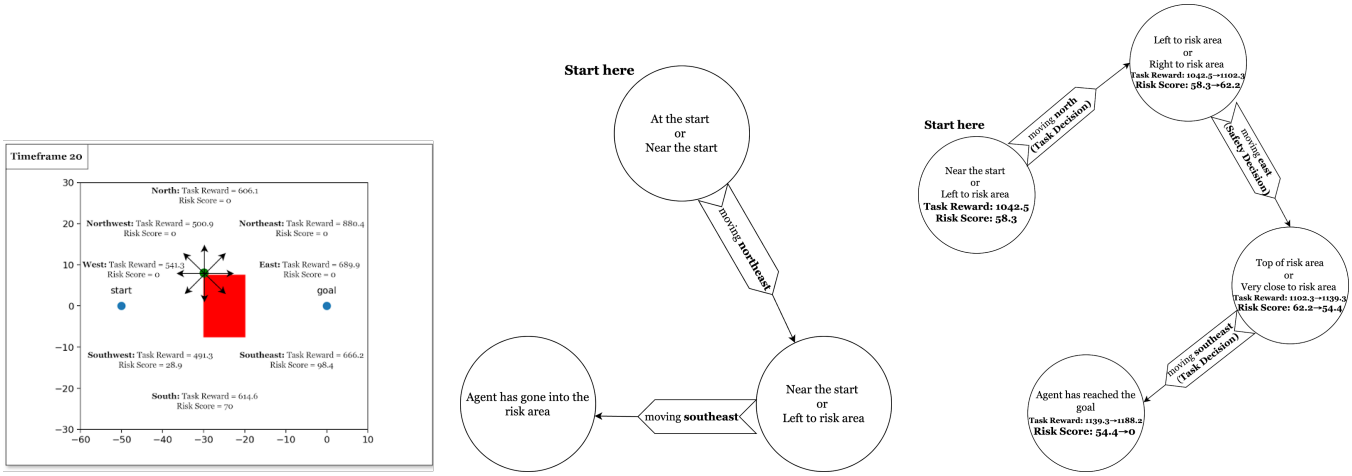
Overview. Fidelity measures how accurately the produced explanation reflect the true behavior of the RL agents. The higher the fidelity, the better alignment between the graph and the agent's behavior, the better the explanation. We calculate fidelity for four components of the agent that xSRL explanations capture: (i) action, (ii) policy selection, (iii) Q_{risk}, and (iv) Q_{task}. The *action* fidelity is measured by the proportion of actions taken by the RL agent that match the *actions* predicted by the produced explanation graph [20]. Similarly, policy selection fidelity measures the extent to which the graph correctly identifies the policy used by the agent to make decisions. Q_{risk} and Q_{task} fidelities measure how closely xSRL's local explanations match the values produced by the agent's policy. We use the weighted average of the Normalized Root Mean Squared Error (NRMSE) to compute the difference between the Q_{risk} and Q_{task} estimates generated by xSRL and those from the agent's policy. A lower NRMSE indicates a better explanation.

To compute all the fidelity scores, we simulate around 2,000 timesteps of agent-environment interactions for several agents: SAC [9] without attacks, SAC under a 50% attack, SAC patched with the safe policy π_{safety} from AdvExRL [26], the safe policy from RRL-MF [33], and the SQRL [31]-trained agent. We generated and averaged fidelity scores across five graphs for each agent.

Table 1: Fidelity scores for the explanations generated by xSRL for different agents: SAC [9] (with and without attacks), and agents patched with AdvExRL [26], RRL-MF [33], and SQRL [31]. A higher action and policy selection ratio indicates better explanation (policy selection is only applicable for AdvExRL [26] and RRL-MF [33]). For Q_{risk} and Q_{task}, a lower NRMSE indicates better fidelity.

	Action	Policy Selection	Q_{risk}	Q_{task}
SAC (w/o attack)	**63.4%**	-	44.5%	30.4%
SAC (w/ attack)	42.8%	-	43.1%	39.2%
advExRL	34.25%	59.2%	33.2%	32.6%
RRL-MF	31.7%	**94.7%**	48.4%	27.4%
SQRL	32.4%	-	62.8%	25.6%

Results and Discussion. Table 1 summarizes the results and highlights several important trends. The SAC agent without attack achieves the highest action ratio fidelity (63.4%), demonstrating that xSRL effectively captures the agent's behavior in a stable environment. However, under attack, the action ratio fidelity for SAC drops to 42.8%, reflecting the agent's reduced consistency under adversarial conditions, which also impacts the accuracy of the explanations. Among the patched agents, AdvExRL performs notably well with a relatively high policy selection ratio (59.2%) and the lowest Q_{risk} NRMSE (33.2%), indicating that AdvExRL provides the most accurate safety explanations. This can be attributed to AdvExRL's training, which maximizes the KL divergence from an optimal adversarial policy [26], making its safety policy more optimal and, therefore, more explainable. RRL-MF achieves the highest policy selection fidelity (94.7%) due to its lower risk estimation, where most actions were taken by the task policy, as reflected in its xSRL graphs (see Appendix C). Confirming that, RRL-MF's Q_{risk} NRMSE is relatively high (48.4%), indicating challenges in estimating risk. This is likely because RRL-MF relies on expert demonstrations for training safe behavior [33], which may not cover all possible risky scenarios, leading to less accurate safety explanations. On the other hand, SQRL excels in task-related explanations with the lowest



(a) Local explanation **(b) Global explanation** **(c) xSRL (local and global) explanation**

Figure 1: Examples of generated explanations for Navigation2 task using our local explanation, global explanation from CAPS [21], and our xSRL that integrates local and global explanations.

Q_{task} NRMSE (25.6%), but it has the highest Q_{risk} NRMSE (62.8%). This discrepancy in risk estimation is likely due to SQRL's joint optimization of both task and safety requirements [31], making it harder to isolate and accurately estimate risk values.

Overall, xSRL provides comprehensive explanations, but its local safety explanations (Q_{risk}) are highly influenced by the safety patching mechanism used. Thus, we recommend using separate policies for safety, as seen in AdvExRL [26], which yields more optimal and interpretable safety explanations.

5.1.2 *User Studies.*

Overview. To empirically assess users' trust in xSRL explanations, we conducted three user studies to evaluate the impact of combining local and global explanations in xSRL, as well as the effect of each of two levels of explanation individually. Each study used three different patched agents (AdvExRL, RRL-MF, and SQRL), resulting in a total of nine distinct conditions. As local explanations apply to specific states, we selected states with the highest Q_{risk} to show users the agent's behavior in risky scenarios.

Procedure. We recruited 270 participants (30 per study), consisting of sophomores from various majors at our institution and participants from Prolific, with IRB approval. We applied specific filters to ensure relevant participant backgrounds. Each participant was randomly assigned to one study to avoid crossover effects. Initially, participants were introduced to Navigation 2 environment, followed by an explanation of the safety constraints, potential attacks on the agent, Q-values, and key study terms (in layperson language). Participants were then shown three videos of the RL agent: one where the agent succeeds without an attack, one where the agent fails under attack, and one where the agent succeeds under attack with a safety patch. After each phase, participants answered four questions across two scenarios: one where the agent was under attack without a safety mechanism and one where the agent was patched with a safety mechanism. The first three questions [1] were: (i) Q1. What action will the agent likely take in a state with high risk? (ii) Q2. Why did the agent choose that action? (iii) Q3. Will the agent succeed (reaching the goal safely)? Q1 evaluates participants' basic comprehension of the explanations and the impact of Q-values (Q_{task}, Q_{risk}). Q2 assesses whether participants can identify the primary motivation behind the agent's action at a given state evaluating whether the decision was made to achieve the goal, avoid a risk area, respond to an adversarial attack. This evaluation provides insight into the participant's comprehension of the agent's motivations rather than an analysis of the action's overall effectiveness which is measured by Q3. Navigation 2 environment was selected because it provides a sufficient level of complexity that would allow participants to focus on understanding the safety aspects of the agent's behavior rather than being distracted by learning the environment itself. Participants were compensated with a base payment of $1.5, plus an additional bonus of 10 cents for each correct answer.

Table 2: H1 Results: Comparison of the average accuracy of participants' answers across all three safe and three unsafe agents using local, global, xSRL explanation methods.

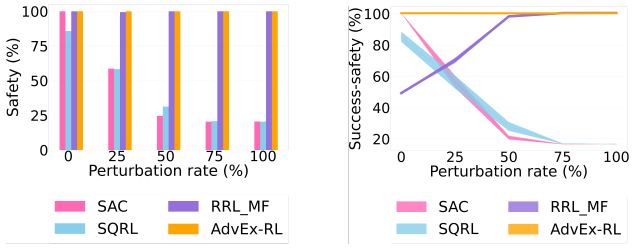
Explanation Method	Unsafe Agents	Safe Agents
Local Explanation	48.14%	68.88%
Global Explanation	**60.37%**	73.7%
xSRL Explanation	**60.37%**	**77.4%**

Hypotheses. We will test two hypotheses:
H1: We hypothesized that using only local or global explanations would be less effective in helping users identify why the agent made certain decisions and whether or not it would succeed under attack, compared to xSRL's combined approach. In other words,

[1] The fourth question is discussed in Section 5.2

Table 3: H2 Results: Comparison of the average accuracy of the participants' answers for three agents patched with AdvExRL, RRL-MF, and SQRL across all explanation methods.

Explanation Method	AdvExRL	RRL-MF	SQRL
Local Explanation	**74.44%**	68.8%	63.3%
Global Explanation	**80%**	40.2%	73.3%
xSRL Explanation	**81.1%**	78.9%	72.20%



(a) Safety (%) **(b) Success-safety (%)**

Figure 2: Safety(%) and success-safety(%) performance of the patched agents under the influence of various rates of the explanation-guided attack in Navigation 2.

participants presented with local or global explanation methods alone will struggle with Q1-Q3 for both safe and unsafe agents, while those presented with xSRL's explanation will perform better.
H2: We hypothesized that agents patched with separate safety policies (AdvExRL and RRL-MF) would be more interpretable than agents using joint optimization like SQRL, where the reasoning behind actions under attack is harder to pinpoint.
Results for H1. Table 2 presents the results for **H1** for both unsafe (under attack agents without safety mechanisms) and safe agents (agents with active safety patches). Results indicate that safe agents were overall more interpretable across all explanation methods, with xSRL providing the highest average accuracy at 77.4%. This is likely because participants could observe Q_{risk} values increasing and then decreasing after actions, which emphasized the agent's safe behavior. In contrast, for unsafe agents, where Q_{risk} remained high, participants found it more difficult to interpret the agent's reasoning, particularly with local explanations (48.14%). Notably, xSRL and global explanations achieved similar performance for unsafe agents, both with an average accuracy of 60.37%. This suggests that the directed graph format used in both methods made it easier for participants to understand the agent's behavior, especially in high-risk scenarios, by providing a broader view of the agent's policy. However, xSRL outperformed both global and local explanations for safe agents, confirming that integrating local and global explanations provides a more comprehensive understanding of the agent's behavior. Overall, xSRL proved to be the *most effective* explanation method, offering more interpretable insights across both safe and unsafe agents to the users, thus confirming H1.

Results for H2. The results from Table 3 reveal important insights regarding the interpretability of the different patched agents (AdvExRL, RRL-MF, and SQRL) across various explanation methods (local, global, and xSRL). AdvExRL consistently shows the highest interpretability across all explanation methods, particularly with xSRL (81.1%). This suggests that its separate safety optimization policy enhances explainability. RRL-MF, while also using a separate safety policy, has slightly lower interpretability (78.9% with xSRL), likely due to challenges in risk estimation. SQRL, which employs joint optimization of task and safety, demonstrates the lowest interpretability (63.3% with local explanations), suggesting that the joint approach makes it harder for participants to grasp the agent's decision-making process, particularly in risk-laden scenarios. This trend reinforces the value of having the safety mechanism separated, as demonstrated by AdvExRL's superior performance across all methods, particularly under attack scenarios, where its safety mechanism is more easily understood by participants. Moreover, global and xSRL explanations overall provided a clearer understanding, specifically for safe agents.

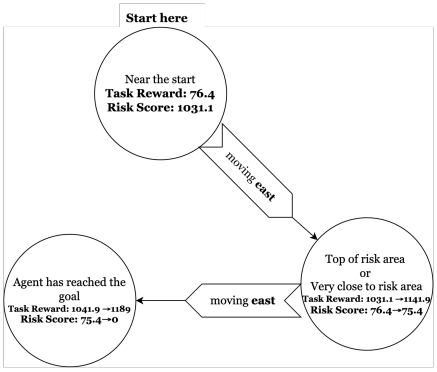
Please refer to Appendix C and D for the detailed results for Navigation 2 and Maze environments. Appendix E has the details of the studies and their design.

5.2 Utility of xSRL's Explanations

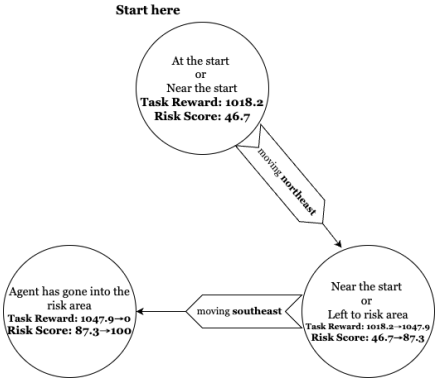
This section evaluates the utility of xSRL's safety explanations in identifying and resolving vulnerabilities in the agent's policy using both computational and empirical approach. The computational approach involves (1) assessing the impact of the explanation-guided attack developed in Section 4.2, and (2) measuring the effectiveness of patching techniques in improving the agent's safety under the same attack. The empirical approach involves analyzing the final question in our user studies to determine whether participants can distinguish between safe and unsafe agents.

5.2.1 Impact of Explanation-guided Attack and Patching Techniques. The explanation-guided attack (AAA) is implemented by targeting abstract states that exhibit higher Q_{risk} values in the xSRL graph of the SAC agent (Figure 3a). We attacked 0, 25, 75 and 100% of the concrete states clustered within these high-risk abstract states. Figure 3b shows the behavior of the SAC agent under attack, without any safety mechanism. To measure the attack's impact, we used two metrics from [26], including *Safety(%)*, which quantifies the proportion of time the agent acts safely over its maximum episode length, and *Success-Safety(%)*, which indicates how close the agent is to reaching its goal. An agent is deemed successful if it finishes the episode within a predefined minimum distance from the goal. For the distance threshold, we used the same value specified in [26] for Navigation2 environment. Figure 2 presents the impact of the attack on SAC without a safety mechanism, as well as patched agents using AdvExRL, RRL-MF, and SQRL, all under 0-100% attack rates. The results clearly show that SAC suffers the most, as its vulnerability is evident in the xSRL graph for its behavior (Figure 3b). In contrast, AdvExRL is the safest, as indicated by the minimal impact of the attack on its safety and success, which is also visually reinforced by its xSRL explanation graph (Figure 3c). Please refer to Appendix C for all remaining results for the agents under attack.

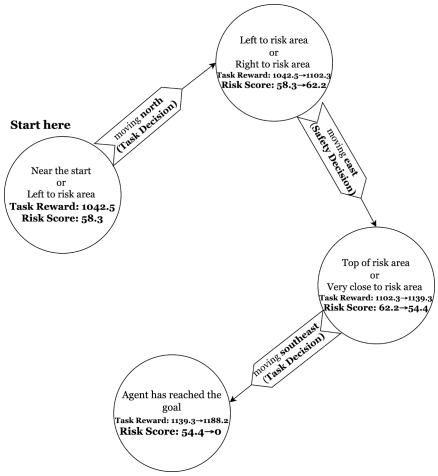
5.2.2 User Studies. In our previous user studies (Section 5.1), we included a subjective question in which participants were asked to identify which agent is safer among two agents, vulnerable and



(a) xSRL explanation of SAC agent *without* attack



(b) xSRL explanation of the SAC agent under attack without a safety patching mechanism (therefore, no task or safety decision labels on the edges).



(c) xSRL explanation of SAC [9] agent under attack with a safety patch

Figure 3: An example of using (a) the xSRL explanation graph to launch an attack on the SAC agent at high-risk states, (b) the xSRL explanation of the agent's behavior under attack, and (c) the explanation of the same agent's behavior under the same attack after being patched with the safe policy from AdvExRL [26].

patched. Participants were also asked to rate their confidence in their choice on a Likert scale from 1 ("not confident at all") to 5 ("very confident"), and we reported the responses with confidence levels of ≥ 4. Both agents were subjected to identical attack scenarios, but *only one* of which was patched with a safety mechanism. To minimize learning effects, participants were allowed to switch between the scenarios and adjust their answers at any point during the study. We hypothesize that if participants had a correct mental model of the agents' strategies, they would be able to identify the safer agent.

Our results show that both xSRL and global explanations achieved comparable results across all patching methods (Table 4). This is likely due to these methods' ability to provide a comprehensive view of the agent's behavior from the start state to the goal state. This allowed participants to observe the final outcomes (success or failure), which directly influenced their choice of the safer agent. Interestingly, despite the results of the objective questions in Table 3 showing that AdvExRL agents were more interpretable, participants expressed higher confidence when evaluating RRL-MF agents in the subjective questions. Overall, participants were able to recognize the safer agents with high confidence across all explanation methods, with confidence levels exceeding 50% in most cases (except for the local explanation of AdvExRL, which was lower). These results reinforce the importance of providing clear and comprehensive safety explanations to aid in understanding behavior of RL agents, particularly in safety-critical environments in which our approach, xSRL, proved to be effective.

6 CONCLUSION

In this paper, we introduced xSRL, a framework that combines local and global explanations to address safety constraints in reinforcement learning (RL) agents. Our approach aims to improve the interpretability of RL policies, with a focus on safety-related behavior, offering developers insights to identify and address vulnerabilities in agent policies. By utilizing policy abstraction and safety-specific visualizations, xSRL provides a clearer understanding of agent decisions and helps users refine policies based on safety concerns. Through computational and empirical evaluations, we showed that xSRL enhances both trust and utility, offering a useful tool for RL policy testing and refinement in safety-critical environments. These results highlight the value of safety-aware explanations in supporting more effective RL system development and user interaction.

Table 4: % of subjects selecting the safer agents across the nine conditions, with confidence ratings ($\geq$4) reported in parentheses.

	ccc		
	Local	**Global**	**xSRL**
AdvExRL	40%(52.6%)	83.33%(62.9%)	70% (54.17%)
RRL-MF	93.33%(72.4%)	90% (44.83%)	90%(62%)
SQRL	76.67%(59.2%)	80%(42.86%)	83.33%(44.44%)

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (NSF) under grant no. 2105007.

REFERENCES

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[3] Dan Amir and Ofra Amir. 2018. HIGHLIGHTS: Summarizing Agent Behavior to People. In *Proceedings of the 17th International Conference on Autonomous Agents*

and MultiAgent Systems (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1168–1176.

[4] Ofra Amir, Finale Doshi-Velez, and David Sarne. 2019. Summarizing agent strategies. *Autonomous Agents and Multi-Agent Systems* 33, 5 (Sept. 2019), 628–644. https://doi.org/10.1007/s10458-019-09418-w

[5] Osbert Bastani. 2021. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *2021 American Control Conference (ACC)*. IEEE, 3488–3494.

[6] Brittany Davis, Maria Glenski, William Sealy, and Dustin Arendt. 2020. Measure Utility, Gain Trust: Practical Advice for XAI Researchers. In *2020 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*. 1–8. https://doi.org/10.1109/TREX51495.2020.00005

[7] Peter Geibel. 2006. Reinforcement learning for MDPs with constraints. In *European Conference on Machine Learning*. Springer, 646–653.

[8] Piyush Gupta, Nikaash Puri, Sukriti Verma, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. [n.d.]. Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution. ([n. d.]). https://par.nsf.gov/biblio/10166401

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[10] Bradley Hayes and J. Shah. 2017. Improving Robot Controller Transparency Through Autonomous Policy Explanation. *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI* (2017), 303–312.

[11] Jacob Hilton, Nick Cammarata, Shan Carter, Gabriel Goh, and Christopher Olah. 2020. Understanding RL vision. https://api.semanticscholar.org/CorpusID:228898185

[12] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2019. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics* 25, 8 (2019), 2674–2693. https://doi.org/10.1109/TVCG.2018.2843369

[13] Sandy H. Huang, David Held, Pieter Abbeel, and Anca D. Dragan. 2019. Enabling robots to communicate their objectives. *Auton. Robots* 43, 2 (feb 2019), 309–326. https://doi.org/10.1007/s10514-018-9771-0

[14] Tobias Huber, Benedikt Limmer, and Elisabeth Andr'e. 2021. Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents. *Frontiers in Artificial Intelligence* 5 (2021). https://api.semanticscholar.org/CorpusID:235490210

[15] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. 2019. Explainable Reinforcement Learning via Reward Decomposition. https://api.semanticscholar.org/CorpusID:204898543

[16] Youngmin Kim, Richard Allmendinger, and Manuel López-Ibáñez. 2020. Safe learning and optimization techniques: Towards a survey of the state of the art. In *International Workshop on the Foundations of Trustworthy AI Integrating Learning, Optimization and Reasoning*. Springer, 123–139.

[17] Isaac Lage, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir. 2019. Exploring computational user models for agent policy summarization. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 1401–1407.

[18] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research* 37, 4-5 (2018), 421–436.

[19] Bing Liu, Yiyuan Xia, and Philip S. Yu. 2004. Clustering Via Decision Tree Construction.

[20] Aniek Markus, Jan Kors, and Peter Rijnbeek. 2021. The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics* 113 (Jan 2021), 103655. https://doi.org/10.1016/j.jbi.2020.103655

[21] Joe McCalmon, Thai Le, Sarra Alqahtani, and Dongwon Lee. 2022. CAPS: Comprehensible Abstract Policy Summaries for Explaining Reinforcement Learning Agents. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 889–897.

[22] Oliver Mihatsch and Ralph Neuneier. 2002. Risk-sensitive reinforcement learning. *Machine learning* 49, 2 (2002), 267–290.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[24] Sina Mohseni, Niloofar Zarei, and Eric D. Ragan. 2018. A Survey of Evaluation Methods and Measures for Interpretable Machine Learning. *ArXiv* abs/1811.11839 (2018). https://api.semanticscholar.org/CorpusID:54087635

[25] Christoph Molnar. 2022. *Interpretable Machine Learning* (2 ed.). https://christophm.github.io/interpretable-ml-book

[26] Md Asifur Rahman, Tongtong Liu, and Sarra Alqahtani. 2023. Adversarial Behavior Exclusion for Safe Reinforcement Learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, 483–491. https://doi.org/10.24963/ijcai.2023/54 Main Track.

[27] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 341–350. https://doi.org/10.1109/TVCG.2016.2599030

[28] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. *CoRR* abs/1502.05477 (2015). arXiv:1502.05477 http://arxiv.org/abs/1502.05477

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *ArXiv* abs/1707.06347 (2017). https://api.semanticscholar.org/CorpusID:28695052

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR* abs/1312.6034 (2013). https://api.semanticscholar.org/CorpusID:1450294

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

[32] Chen Tessler, Yonathan Efroni, and Shie Mannor. 2019. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*. PMLR, 6215–6224.

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

[34] Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Felix Li, Rowan McAllister, Joseph E Gonzalez, Sergey Levine, Francesco Borrelli, and Ken Goldberg. 2020. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3612–3619.

[35] Nicholay Topin and Manuela Veloso. 2019. Generation of Policy-Level Explanations for Reinforcement Learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2514–2521. https://aaai.org/ojs/index.php/AAAI/article/view/4097

[36] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

[37] Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, and Mark Antonius Neerincx. 2018. Contrastive Explanations for Reinforcement Learning in terms of Expected Consequences. *ArXiv* abs/1807.08706 (2018). https://api.semanticscholar.org/CorpusID:49907182

[38] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. *CoRR* abs/1509.06461 (2015). arXiv:1509.06461 http://arxiv.org/abs/1509.06461

[39] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B. Viégas, and Martin Wattenberg. 2018. Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 1–12. https://doi.org/10.1109/TVCG.2017.2744878

[40] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. 2016. Graying the black box: Understanding DQNs. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1899–1908. https://proceedings.mlr.press/v48/zahavy16.html