Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking

Anonymous authors
Paper under double-blind review

Abstract

Ensuring the safety of reinforcement learning (RL) algorithms is crucial to unlock their potential for many real-world tasks. However, vanilla RL and most safe RL approaches do not guarantee safety. In recent years, several methods have been proposed to provide hard safety guarantees for RL, which is essential for applications where unsafe actions could have disastrous consequences, e.g., a car crash. Nevertheless, there is no comprehensive comparison of these provably safe RL methods. Therefore, we introduce a categorization of existing provably safe RL methods, present the conceptual foundations for both continuous and discrete action spaces, and empirically benchmark existing methods. We categorize the methods based on how they adapt the action: action replacement, action projection, and action masking. Our experiments on an inverted pendulum and a quadrotor stabilization task indicate that action replacement is the best-performing approach despite its comparatively simple formulation. Furthermore, adding a reward penalty every time the safety verification is engaged improves training performance. Finally, we provide practical guidance on selecting provably safe RL approaches depending on the safety specification, RL algorithm, and type of action space.

1 Introduction

Reinforcement learning (RL) contributes to many recent advancements in challenging research fields such as robotics (El-Shamouty et al., 2020; Zhao et al., 2020), autonomous systems (Kiran et al., 2021; Ye et al., 2021), and games (Mnih et al., 2013; Silver et al., 2017). A vanilla RL agent typically explores at random and usually executes undesired actions multiple times to learn how to achieve the highest possible reward. However, for many applications safety is important. Therefore, safe RL emerged where the learning process is adapted such that the agent considers safety aspects next to performance during training and/or operation (García & Fernández, 2015). There are different degrees of how safety is considered for safe RL approaches. First, there are approaches that only incorporate safety aspects without formal guarantees. Here, the agent chooses safe actions with higher probability, e.g., by adding a reward component that indicates risk, or adapting the exploration such that the agent follows a safe heuristic. Second, there are approaches that provide probabilistic safety guarantees, e.g., by using probabilistic models for action safety (Könighofer et al., 2021). Still, whenever failures are disastrous and need to be avoided at all costs during training and deployment, hard safety guarantees for the RL agent are necessary. This is the case for safety-critical applications such as autonomous driving, human-robot collaboration, or energy grids. We refer to this third subcategory of safe RL methods that provide hard safety guarantees for both training and operation as *provably safe RL*.

In this paper, we provide for the first time a consistent conceptual framework for *provably safe RL* in both continuous and discrete action spaces, a comprehensive literature survey, and a comparison between provably safe RL approaches on two widely-used control benchmarks. The characteristic difference between provably safe RL approaches is how they adapt the actions of the agent. Therefore, we propose classifying them into three categories: *action replacement*, *action projection*, and *action masking*. Our proposed categorization of provably safe RL provides a concise presentation of the research field, supports researchers implementing provably safe RL through a clear terminology and comprehensive literature review, and outlines ideas for future research within the three categories. We, furthermore, evaluate the methods experimentally with the three main findings that all provably safe RL methods are indeed safe, that action replacement performs best on average over five tested RL algorithms, and that adding a penalty to the reward when using the safety function further improves performance.

Our contributions in this work are fourfold. First, we introduce a comprehensive classification of provably safe RL methods and their formal description. This categorization allows us to compare and benchmark the effects of choosing a specific type of action modification on the ability of agents to learn. Second, we propose the first formulation of action masking for continuous action spaces. Third, we provide a structured and comprehensive survey of previous provably safe RL works and assign them to the three categories. Finally, we are the first to evaluate the performance of all three provably safe RL methods on two common control benchmarks. This comparison provides insights into the strengths and weaknesses of the different provably safe RL approaches and allows us to provide advice on selecting the best-suited provably safe RL approach for a specific problem independent of the safety verification method used.

The remainder of this paper is structured as follows. First, we briefly overview the historical development of safe RL and relate provably safe RL to it in section 1.1. We describe preliminary concepts in section 2 and introduce our proposed categorization. We then show how the related provably safe RL literature fits our categorization in section 3. Section 4 compares the different provably safe RL categories experimentally on a two-dimensional (2D) quadrotor stabilization task. Section 5 discusses the results of our experimental evaluation and the practical considerations following them. Finally, we conclude this work in section 6.

1.1 Evolution towards provably safe RL

The notion of risk and safety in RL has existed since the 1990s (Heger, 1994). The reasons for combining safety and RL were to focus the learning on relevant or safe regions and improve the convergence speed. Thus, the field of safe RL started developing, and in 2015, García & Fernández (2015) were the first to cluster safe RL. They provide two high-level categories: approaches that modify the optimization criterion and approaches that modify the exploration. Since 2015 significant advances in model-free RL and the increased applicability of deep RL changed the research focus of safe RL. Importantly, the higher efficiency of model-free deep RL made real-world application tangible and amplified the need for formal guarantees in safe RL. This is also apparent from the survey by Brunke et al. (2022), who investigate recent developments at the intersection of control and learning for safe robotics. As goal of the broader safe learning for control field they identify methods with as little as possible system knowledge and formal safety guarantees (Brunke et al., 2022, Fig. 4). Among existing safe RL approaches, provably safe RL research is a growing field located at this frontier as it provides hard safety guarantees during both learning and deployment. While a few papers mentioned in Brunke et al. (2022) are part of provably safe RL, it is not a focus of their work. In the following paragraphs, we use the common classification by safety specification type, which can be *soft constraints*, *probabilistic*, and *hard guarantees*, to locate provably safe RL in the field of safe RL.

Soft constraints Soft constraint approaches consider safety directly in their optimization objective. Here, the agent can explore all actions and states regardless of safety. Thus, these methods can be unsafe during training, especially in the beginning, but converge to a safer policy without formal safety guarantees after sufficient training steps. The simplest way to inform an RL agent about safety constraints is through its reward function. Despite its elegance, the reward function approach has many potential pitfalls. First, the reward function might be ill-defined, either from manual tuning or when learned from human input. When manually defined, the reward function might overlook certain features or fine details, leading to a hackable reward (Skalse et al., 2022) from which the agent learns an unsafe behavior. Learning the reward function from human feedback (Christiano et al., 2017) is also error-prone because communicating safety constraints alongside performance metrics is hard for sparse, non-linear, conditional, or seldom occurring constraints. Second, even if the reward function is defined correctly, the trained policy is not guaranteed to be safe, as it was shown by Packer et al. (2019) that RL agents struggle with out-of-distribution states during deployment. Third, the agent might learn to perform actions safely but ignore the task objective due to goal misgeneralization (Langosco et al., 2022). A more complicated way to inform an RL agent is formulating a constraint optimization problem. Many recent advances have been made in constrained RL (Altman, 1998;

Achiam et al., 2017; Stooke et al., 2020), for which the policy aims to maximize the reward while satisfying user-defined specifications. The specifications can be formulated as constraint functions (Chow et al., 2018; Stooke et al., 2020; Yang et al., 2020; Marvi & Kiumarsi, 2021) or as temporal logic formulas (De Giacomo et al., 2021; Hasanbeig et al., 2020; 2019b;a). The main advantage of these methods is that no explicit model of the agent dynamics or the environment is required as the agent learns the safety aspects through experience. Thus, such safe RL methods have a high potential in non-critical settings, where unsafe actions do not cause major damage.

Probabilistic guarantees Probabilistic safe RL approaches rely on probabilistic models or synthesize a model from sampled data. Here, the action and state space can be restricted based on probabilities. Nonetheless, unsafe actions are sometimes not detected and might occur occasionally. Several works (Turchetta et al., 2016; Berkenkamp et al., 2017; Mannucci et al., 2018) try to determine the maximal set of safe states by starting from an often user-defined conservative set and extending it with the gathered learning experience. Other methods (Könighofer et al., 2021; Thananjeyan et al., 2021; Dalal et al., 2018; Zanon & Gros, 2021; Yang et al., 2021; Gillula & Tomlin, 2013) are based on formulating probabilistic models that identify the probability of safety for an action. In general, approaches that rely on probabilistic methods are especially applicable if one cannot bind measurement errors, modeling errors, and disturbances by sets.

Hard guarantees Provably safe RL defines hard safety guarantees, which are fulfilled by integrating prior system knowledge into the learning process. Here, the agent only explores safe actions and only reaches states fulfilling the safety specifications. Provably safe RL already fulfills the given safety specifications during the learning process, which is essential when training or fine-tuning agents on safety-critical tasks in the physical world. Thus, we exclude approaches that only verify learned policies (Bastani et al., 2018; Schmidt et al., 2021) from our survey. We focus on model-free RL algorithms that do not explicitly learn or use a model of the system dynamics to optimize the policy. Generally, deploying learned controllers in the physical world became increasingly realistic in recent years, and thus, the need for provably safe RL grew, and more provably safe RL approaches were developed. With this work, we aim to structure and provide practical insights into this growing field.

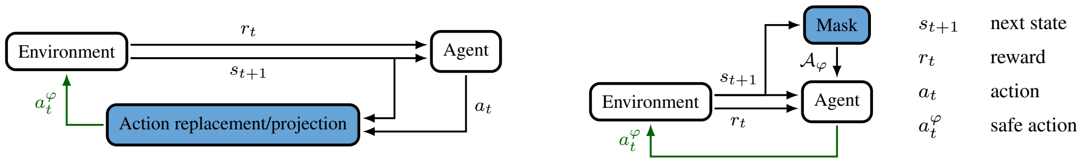
2 Conceptual Analysis

We introduce three provably safe RL classes by providing their formal notation in one comprehensive conceptual framework. This framework clarifies the differences between the three classes and eases the following literature review and benchmarking.

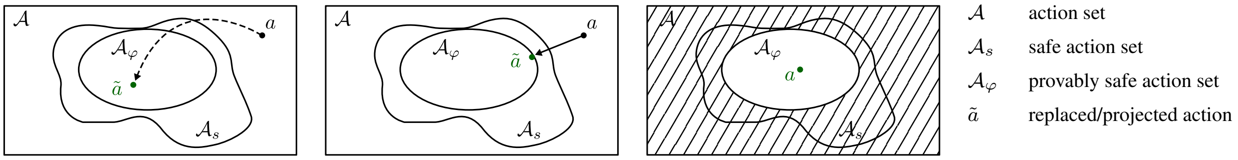
Markov decision process The RL agent learns on a Markov decision process (MDP) that is described by the tuple $(\mathbb{S}, \mathbb{A}, T, r, \gamma)$. Hereby, we assume that the set of states $\mathbb{S}$ is fully observable with bounded precision. Partially observable MDPs can be handled using methods like particle filtering (Sunberg & Kochenderfer, 2018) and are not further discussed in this work. Both the action space $\mathbb{A}$ and state space $\mathbb{S}$ can be either continuous or discrete. $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ is the transition function, which in the discrete case gives the probability that the transition from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ occurs by taking action $\boldsymbol{a}$. In the continuous case, $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ denotes the probability density function of the transition. We assume that the transition function is stationary over time. For each transition, the agent receives a reward $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ from the environment. Finally, the discount factor $0 < \gamma < 1$ weights the relevance of future rewards.

Safety of a system For provably safe RL, it is required that the safety of states and actions are verifiable. Otherwise, no formal claims about the safety of a system can be made. Thus, we first introduce the set of provably safe states $\mathbb{S}_\varphi \in \mathbb{S}$ that contains all states in which all safety constraints are fulfilled. Note that the state space is often augmented from the classical control state space to a state space that also includes other safety-relevant dimensions, e.g., action space with constraints. For verifying the safety of actions, we use a safety function $\varphi : \mathbb{S} \times \mathbb{A} \to \{0, 1\}$

$$\varphi(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 1, & \text{if } (\boldsymbol{s}, \boldsymbol{a}) \text{ is verified safe} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$



(a) Post-posed action replacement / projection (b) Preemptive action masking



(c) Action replacement (d) Action projection (e) Action masking

Figure 1: Structure of the three types of provably safe RL methods. The post-posed action replacement or projection methods (a) alter unsafe actions before sending them to the environment. In contrast, preemptive action masking approaches (b) allow the agent to only choose from the safe action space and therefore only output safe actions to the environment. Figures (c-e) highlight the differences of the three approaches in the action space. Here, action replacement (c) replaces unsafe actions with actions from the safe action space, action projection (d) projects unsafe actions to the closest safe action, and action masking (e) lets the agent choose solely from the safe action set.

Conceptually, this is mostly done by over-approximating the set of states that are reachable by taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$, and then validating if the reachable set of states is a subset of $\mathbb{S}_\varphi$. We discuss the concrete verification methods used by previous works in section 3. It is clear that $\varphi(\boldsymbol{s}, \boldsymbol{a})$ requires some prior knowledge about the system dynamics. We can provide this knowledge with an abstract model that contains all safety-relevant behaviors of the original model. These abstract safety models can get different inputs than the RL agent, and are in general less complex than the underlying MDP. In systems where such a safety model is unavailable, provably safe RL is not applicable, and only non-provably safe approaches as discussed in section 1 can be used. Although $\varphi(\boldsymbol{s}, \boldsymbol{a})$ only verifies one action $\boldsymbol{a}$, it may take more than the next state into account, e.g., through an model predictive control (MPC) formulation.

Based on the safety function, we define a set of provably safe actions $\mathbb{A}_\varphi(\boldsymbol{s}) = \{\boldsymbol{a} | \varphi(\boldsymbol{s}, \boldsymbol{a}) = 1\}$ for a state $\boldsymbol{s}$, which is a subset of all safe actions $\mathbb{A}_{\boldsymbol{s}}(\boldsymbol{s})$, i.e., $\mathbb{A}_\varphi(\boldsymbol{s}) \subseteq \mathbb{A}_{\boldsymbol{s}}(\boldsymbol{s}) \subseteq \mathbb{A}$[1]. The safe action set $\mathbb{A}_{\boldsymbol{s}}(\boldsymbol{s})$ includes all safe actions while the provably safe action set $\mathbb{A}_\varphi(\boldsymbol{s})$ only includes actions that are verified as safe by the safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$. In other words, the safety function possibly returns that an action is unsafe which is indeed safe, while it never predicts truly unsafe actions to be safe. All provably safe RL approaches rely on the availability of provably safe actions, so we require assumption 1.

Assumption 1 *There is at least one provably safe initial state* $\boldsymbol{s}_0^\varphi \in \mathbb{S}_\varphi$ *and for all safe states there exits at least one safe action* $\forall \boldsymbol{s}^\varphi \in \mathbb{S}_\varphi \rightarrow \mathbb{A}_\varphi(\boldsymbol{s}^\varphi) \neq \emptyset$*.*

With assumption 1, it is ensured that only provably safe states $\boldsymbol{s}^\varphi \in \mathbb{S}_\varphi$ can be reached when starting from any $\boldsymbol{s}_0^\varphi$ and taking only provably safe actions thereafter.

Provably safe RL relies on model knowledge to provide safety guarantees, i.e., a conformant model that covers the safety-relevant system and environment dynamics. Hereby, the verification process can use an abstraction of the real system as long as it is conformant (Roehm et al., 2019; Liu et al., 2023) to the real system, i.e., it over-approximates both aleatoric and epistemic uncertainties, and covers all relevant safety aspects. This eases efficient verification, as the complexity of the abstraction is usually significantly lower than the complexity of the real system. In practice, the safety specifications are often weakened to legal or passive safety. Hereby, inevitable safety violations caused by other agents are not considered to be the fault

---

[1] Please note that “taking no action” is commonly considered to be part of the action space, most often with the action $\boldsymbol{a} = [0, \dots, 0]^\top$.

of the agent and are therefore not considered unsafe. Examples of proving legal safety have been presented for autonomous driving (Pek et al., 2020) and in robotics (Bouraine et al., 2012).

There are multiple ways to ensure provable safety for RL systems, which we summarize in the three categories: action replacement, where the safety method replaces all unsafe actions from the agent with safe actions, action projection, which projects unsafe actions to the safe action space, and action masking, where the agent can only choose actions from the safe action space. We choose this categorization as it represents the three main approaches found in the literature to modify actions and thereby ensure safety for RL. Action replacement and action projection alter the action after it is outputted by the agent, so we refer to them as post-posed methods. Whereas, action masking only lets the agent choose from the safe action space; it is therefore a preemptive safety measure. Figure 1 displays the basic concept and structure of these methods. The following subsections describe the concept, mathematical formalization, required assumptions, and practical implications of the three approaches.

2.1 Action replacement

The first approach to ensure the safety of actions is to replace any unsafe action outputted by the agent with a safe action before its execution. The first step of action replacement is to evaluate the safety of the suggested action $\boldsymbol{a} \in \mathbb{A}$ using $\varphi(\boldsymbol{s}, \boldsymbol{a})$. If the action sampled from the policy $\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s})$ is not verified as safe, it is replaced with a provably safe replacement action $\tilde{\boldsymbol{a}} = \boldsymbol{\psi}(\boldsymbol{s})$, where $\boldsymbol{\psi} : \mathbb{S} \rightarrow \mathbb{A}_\varphi$ is called replacement function. Following this procedure, it is guaranteed that only safe actions $\boldsymbol{a}^\varphi$ with

$$\boldsymbol{a}^\varphi = \begin{cases} \boldsymbol{a} \sim \boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s}), & \text{if } \varphi(\boldsymbol{s}, \boldsymbol{a}) = 1 \\ \boldsymbol{\psi}(\boldsymbol{s}), & \text{otherwise} \end{cases} \tag{2}$$

are executed. We discuss how this action replacement alters the MDP in the Appendix and additionally refer the interested reader to Hunt et al. (2021).

There are two general replacement functions found in literature, *sampling* and *failsafe*. In sampling, the replacement function $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ uniformly samples a random action from $\mathbb{A}_\varphi(\boldsymbol{s})$. The other approach is to use a backup failsafe controller $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ as replacement action, which could also stem from human feedback. In time-critical and complex scenarios, where building $\mathbb{A}_\varphi(\boldsymbol{s})$ online becomes too time-consuming, $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ is the only available option.

2.2 Action projection

In contrast to action replacement, where the replacement action is not necessarily related to the agent's action, action projection returns the closest provably safe action with respect to the original action and a distance function. For this, we define the optimization problem

$$\begin{aligned} \arg\min_{\tilde{\boldsymbol{a}}} \quad & \operatorname{dist}(\boldsymbol{a}, \tilde{\boldsymbol{a}}) \\ \text{subject to} \quad & \varphi(\boldsymbol{s}, \tilde{\boldsymbol{a}}) = 1, \end{aligned} \tag{3}$$

where dist$(\cdot)$ describes an arbitrary distance function, e.g., any p-norm. Note, that it might not be possible to define such a distance function, especially in discrete action spaces. The constraints are often defined explicitly by constraint functions $f_i(\tilde{\boldsymbol{a}}, \boldsymbol{s}) \leq 0$, $\forall\, i \in 1, \ldots, n$ that describe that the next state $\boldsymbol{s}' \in \mathbb{S}_\varphi$. The optimization problem in (3) minimizes the alteration of the actions while satisfying the safety constraints. Following assumption 1, we require that the optimizer always finds a solution for the problem in (3) if any exists.

The most prominent ways to formulate the safety constraints for action projection are based on control barrier functions (CBFs) or robust MPC. For the first method, the constraints are defined by CBFs (Wieland & Allgöwer, 2007) that translate state constraints to control input constraints. We formulate the CBFs according to Taylor et al. (2020) as it is an intuitive formulation for RL. Given is a nonlinear control-affine system

$$\dot{\boldsymbol{s}} = \boldsymbol{m}(\boldsymbol{s}) + \boldsymbol{b}(\boldsymbol{s})\, \tilde{\boldsymbol{a}}, \tag{4}$$

where $\boldsymbol{s} \in \mathbb{S} \subseteq \mathbb{R}^N$ is the continuous state with N dimensions, and $\tilde{\boldsymbol{a}} \in \mathbb{A} \subset \mathbb{R}^M$ is the continuous control input with M dimensions and $\boldsymbol{m}(\boldsymbol{s})$ and $\boldsymbol{b}(\boldsymbol{s})$ are locally Lipschitz continuous functions. Then, the function h is a CBF if there exists an extended class $\mathbb{K}$ function α such that (Taylor et al., 2020, Eq. 3)

$$\dot{h}(\boldsymbol{s}, \tilde{\boldsymbol{a}}) \geq -\alpha(h(\boldsymbol{s})). \tag{5}$$

The limitation to control-affine systems makes the formulation of the constrained optimization problem efficient, e.g., for a Euclidean norm as distance function, (3) results in a quadratic programming (QP). A downside of using CBFs is that $h(\boldsymbol{s})$ is not trivial to find, especially in environments with dynamic obstacles.

For the second common projection method, we formulate the optimization problem with MPC according to Wabersich & Zeilinger (2021). There, input constraints are formulated next to the system dynamics and safety constraints, which are usually given as state constraints. The constraints in (3) can then be formulated to find an input trajectory that steers the system from the current state $\boldsymbol{s}$ to the safe terminal set $\mathbb{M} \subseteq \mathbb{S}_\varphi$ within the prediction horizon L (Wabersich & Zeilinger, 2021, Eq. 5):

$$\begin{aligned} \underset{\tilde{\boldsymbol{a}}}{\arg\min} \quad & \operatorname{dist}(\boldsymbol{a}, \tilde{\boldsymbol{a}}) \\ \text{subject to} \quad & \boldsymbol{s}_{l+1} = \boldsymbol{g}(\boldsymbol{s}_l, \tilde{\boldsymbol{a}}_l, \boldsymbol{d}), \boldsymbol{s}_0 = \boldsymbol{s}, \\ & \boldsymbol{s}_l \in \mathbb{S}_\varphi \; \forall\, l \in \{1, ..., L-1\}, \\ & \boldsymbol{s}_L \in \mathbb{M}, \\ & \boldsymbol{a}_l \in \mathbb{A} \; \forall\, l \in \{1, ..., L\}, \\ & \tilde{\boldsymbol{a}} = \boldsymbol{a}_1, \end{aligned} \tag{6}$$

where $\boldsymbol{s}_l$ and $\boldsymbol{a}_l$ are the state and action l-steps ahead of the current time step, and $\boldsymbol{g}(\cdot)$ describes the system dynamics including disturbances $\boldsymbol{d}$. For the set $\mathbb{M}$, a controller exists that keeps the agent within this set for an infinite time. If the optimization problem is solvable, then $\tilde{\boldsymbol{a}}$ is executed. If it is not solvable, the control sequence from the previous state is used as a backup plan until the safe terminal set is reached or the optimization problem is solvable again (Schürmann et al., 2018). The MPC formulation allows one to easily integrate model and measurement uncertainties. However, for an environment with dynamic obstacles, the safe terminal set can be time-dependent and we are not aware of a straightforward integration that still guarantees assumption 1.

For both CBF and MPC approaches, nominal models are used to model the known system dynamics. The part of the system dynamics that are unknown can be modelled as disturbances

$$\dot{\boldsymbol{s}} = \boldsymbol{m}(\boldsymbol{s}) + \boldsymbol{b}(\boldsymbol{s})\, \tilde{\boldsymbol{a}} + \boldsymbol{d}, \tag{7}$$

where safety can be ensured if $\boldsymbol{d}$ is a time signal that is essentially bounded in time $t \geq 0$, $d(t) \leq \bar{d} > 0$ for every entry d of $\boldsymbol{d}$ and where the bound is finite $\bar{d} < \inf$ (Taylor et al., 2021). To reduce conservatism, disturbances can be modelled as state and input depended $\boldsymbol{d}(\boldsymbol{s}, \tilde{\boldsymbol{a}})$ and the maximal disturbance can be learned from data as presented in Taylor et al. (2021) for CBF, and Hewing et al. (2020) for MPC models.

2.3 Action masking

The two previous approaches modify unsafe actions from the agent. In action masking, we do not allow the agent to output an unsafe action in the first place (preemptive method). Hereby, a mask is added to the agent so that it can only choose from actions of the provably safe action set. In addition to assumption 1, action masking in practice requires an efficient function $\boldsymbol{\eta}(\boldsymbol{s}) : \mathbb{S} \rightarrow \mathbb{A}$ that determines a sufficiently large set of provably safe actions $\mathbb{A}_\varphi \subseteq \mathbb{A}$ for a given state $\boldsymbol{s}$. The policy function $\boldsymbol{\pi}$ is informed by the function $\boldsymbol{\eta}(\boldsymbol{s})$ and the action selection is adapted such that only actions from $\mathbb{A}_\varphi$ can be selected:

$$\boldsymbol{a} \sim \boldsymbol{\pi}(\mathrm{a} | \boldsymbol{\eta}(\boldsymbol{s}), \boldsymbol{s}) \in \mathbb{A}_\varphi. \tag{8}$$

If $\boldsymbol{\eta}(\boldsymbol{s})$ can only verify one or a few actions efficiently, the agent cannot learn properly because the agent cannot explore different actions and find the optimal one among them. Ideally, the function $\boldsymbol{\eta}(\boldsymbol{s})$ achieves $\mathbb{A}_\varphi = \mathbb{A}_s$.

The action masking approaches for discrete and continuous action spaces are not easily transferable into each other, and will therefore be discussed separately in this subsection. For discrete actions, the safety of each action is verified in each state using $\varphi(\boldsymbol{s}, \boldsymbol{a})$ and all verified safe actions are added to $\mathbb{A}_{\varphi}(\boldsymbol{s})$, i.e., $\boldsymbol{\eta}$ iterates over all actions for the current state $\boldsymbol{s}$ with $\varphi(\boldsymbol{s}, \boldsymbol{a})$ to identify $\mathbb{A}_{\varphi}(\boldsymbol{s})$. Intuitively, the discrete action mask is an informed drop-out layer added at the end of the policy network. We define the resulting safe policy $\boldsymbol{\pi}_m(\boldsymbol{a}|\boldsymbol{s})$ as

$$\boldsymbol{\pi}_m(\boldsymbol{a}|\boldsymbol{s}) = \varphi(\boldsymbol{s}, \boldsymbol{a}) \frac{\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s})}{\sum_{\boldsymbol{a}^{\varphi} \in \mathbb{A}_{\varphi}(\boldsymbol{s})} \boldsymbol{\pi}(\boldsymbol{a}^{\varphi}|\boldsymbol{s})}. \tag{9}$$

The integration of masking in a specific learning algorithm is not trivial. The effects on policy optimization methods are discussed in Krasowski et al. (2020); Huang & Ontañón (2022). For RL algorithms that learn the Q-function, we exemplary discuss the effects of discrete action masking for deep Q-network (DQN) (Mnih et al., 2013), which is most commonly used for Q-learning with discrete actions. During exploration with action masking, the agent samples its actions uniformly from $\mathbb{A}_{\varphi}$. When the agent exploits the Q-function, it chooses only the best action among $\mathbb{A}_{\varphi}$, i.e., $\arg\max_{\boldsymbol{a}} Q(\boldsymbol{s}, \boldsymbol{a})$. The temporal difference error for updating the Q-function $Q(\boldsymbol{s}, \boldsymbol{a})$ is (Mnih et al., 2013, Eq. 3)

$$r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a}), \tag{10}$$

where the action in the next state is $\boldsymbol{a}' \in \mathbb{A}_{\varphi}$ in contrast to the vanilla temporal difference error where the maximum Q-value for the next state is searched among actions from $\mathbb{A}$. Using the adapted temporal difference error in (10), the learning updates are performed only with Q-values of actions relevant in the next state instead of the full action space.

To comprehensively compare the different provably safe RL on discrete and continuous action spaces in section 4, we developed a formulation for continuous masking since there is no existing research. We formulate continuous action masking as a transformation of the action of agents to the provable safe action set. The action space $\mathbb{A}$ is transformed into $\mathbb{A}_{\varphi}$ by applying the transformation

$$\tilde{\boldsymbol{a}} = (\boldsymbol{a} - \min(\mathbb{A})) \frac{\max(\mathbb{A}_{\varphi}) - \min(\mathbb{A}_{\varphi})}{\max(\mathbb{A}) - \min(\mathbb{A})} + \min(\mathbb{A}_{\varphi}) \tag{11}$$

to the actions $\boldsymbol{a}$, where $\mathbb{A}_{\varphi}$ is assumed to be convex, $\min(\cdot)$ and $\max(\cdot)$ return a vector containing the minimal and maximal value of the given set in each dimension respectively, and all operations are evaluated element-wise. For example, given a two dimensional continuous action space $\mathbb{A} = [0, 1] \times [-1, 2]$, then $\min(\mathbb{A}) = [0, -1]^{\top}$. Note that this implementation approximates the provably safe action set with an interval set which can be conservative. To overcome this limitation of this first formulation of continuous action masking, more complex set representations, such as the zonotopes (Althoff et al., 2021), for the action spaces ($\mathbb{A}$ and $\mathbb{A}_{\varphi}$) in combination with solving an optimization problem that maximizes the size of $\mathbb{A}_{\varphi}$ could be investigated. A less sophisticated yet possibly effective approach is searching for a good latent interval representation of and transformation to $\mathbb{A}_{\varphi}$ by applying principal component analysis to a set of $\mathbb{A}_{\varphi}$ for different states as a pre-computing step. The necessary properties of this transformation are discussed in the next paragraph.

Since the action spaces for RL are defined a priori, there must always be a valid transformation from $\boldsymbol{a}$ to $\tilde{\boldsymbol{a}}$ such that $\tilde{\boldsymbol{a}} \in \mathbb{A}_{\varphi}$ and such that the operation is deterministic and time-invariant for a specific state and action pair. In contrast to action replacement, the action is only transformed rather than replaced with a potentially non-deterministic safe action. In this way the agent can explore the full action space $\mathbb{A}$ since (11) ensures that the action is transformed to the safe mapping of it. If one simply clips the action to the safe action space, the policy distribution is altered so that all probabilities for actions outside the clip range are projected to the safe action on the boundary of the safe action space. Thus, the probability distribution differs from the distribution of the current policy, which can be problematic for on-policy RL. Additionally, actions on the bound of the provably safe action space become disproportional more likely. In contrast, the action transformation we propose in (11) conserves the original policy and is conceptually similar to action normalization, which is commonly used in RL (Sutton & Barto, 2018, Ch. 16.8).

2.4 Learning tuples

When changing the RL action, the training of the agent can be conducted with four possible learning tuples:

- *naive* - learning based on the action outputted by the policy network of the agent $\boldsymbol{a}$ and the reward $r(\boldsymbol{s}, \boldsymbol{a}^{\varphi})$ corresponding to the executed action $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r)$. This ensures that the agent is updated according to its current policy. Learning with the original action should benefit on-policy learning, where the policy is updated based on experience collected using the most recent policy. However, utilizing the *naive* tuple for training is subject to state transitions that rely heavily on the replacement function $\boldsymbol{\psi}(\boldsymbol{s})$, as demonstrated in the Appendix. This can be viewed as a form of noise on the observation-action relationship, thereby possibly hindering effective learning.
- *adaption penalty* - is *naive* with a penalty $r^*(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{a}^{\varphi}) = r(\boldsymbol{s}, \boldsymbol{a}^{\varphi}) + r_{\text{penalty}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{a}^{\varphi})$ if an unsafe action was selected $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r^*)$. In action projection, the penalty r_{penalty} can also include a term that is proportional to the projection distance $\text{dist}(\boldsymbol{a}, \tilde{\boldsymbol{a}})$, as proposed by Wang (2022).
- *safe action* - learning based on the safe and possibly adapted action and the corresponding reward $(\boldsymbol{s}, \boldsymbol{a}^{\varphi}, \boldsymbol{s}', r)$. By using the *safe action* tuple, we are correctly rewarding the agent for the actual performed transition. However, this requires updating the agent with an action that did not stem from the agent's current policy $\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s})$. This is an expected behavior in off-policy learning, so it is assumed that the safe action tuple is a better fit for off-policy than for on-policy RL.
- *both* - in case the RL agent proposes an unsafe action, both the *adaption penalty* and the *safe action* tuples are used for learning, otherwise *naive* is used.

In all cases, the next state $\boldsymbol{s}'$ and reward r are the true state and reward received from the environment after executing the safe action $\boldsymbol{a}^{\varphi}$.

Action masking is in the literature always paired with the *naive* or *adaption penalty* learning tuple. The *adaption penalty* is related to the reduction of the action space due to the masking or a safety component in the reward function and not a sparse reward as for action projection and action replacement. For discrete action masking, there is no *safe action* tuple since there is no $\tilde{\boldsymbol{a}}$ (see figure 1). For continuous action masking, using the *safe action* tuple with the transformed action $\tilde{\boldsymbol{a}}$ leads to inconsistencies in training. For example, the agent chooses in state $\boldsymbol{s}_1$ the action $\boldsymbol{a}_1$ which is transformed to $\tilde{\boldsymbol{a}}_1$. If the agent visits state $\boldsymbol{s}_1$ again, and chooses $\boldsymbol{a}_2 = \tilde{\boldsymbol{a}}_1$, the executed action becomes $\tilde{\boldsymbol{a}}_2 \neq \boldsymbol{a}_2$ due to the transformation.

3 Literature Review

In this section, we summarize previous works in provably safe RL and assign them to the proposed categories. To identify the related literature, we used the search string `TITLE-ABS("reinforcement learning") AND TITLE(learning) AND [TITLE(safe*) OR TITLE(verif*) OR TITLE(formal*) OR TITLE( shield*)] AND LIMIT-TO(LANGUAGE,"English")` for the Scopus[2] and IEEEXplore[3] search engine, which led to 620 papers already removing duplicates. Then, we screened papers by title and abstracts to identify 160 seemingly relevant papers. After close inspection, we identified 47 of these 160 papers as provably safe RLworks. We give a condensed overview of all application-independent provably safe RL works in Table 1, and cluster all 47 provably safe RL works in Table 2 by their application.

Action replacement One of the earliest provably safe RL works are Alshiekh et al. (2018), which construct a so-called safety shield from linear temporal logic formulas. Hereby, they construct the verification function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ by converting the linear temporal logic formulas into an automaton, on which they then perform model checking. The advantage of online model checking is that linear temporal logic constraints can be guaranteed for general nonlinear systems and the online complexity is linear in state dimensions. However, the method is only applicable to discrete state spaces, constructing the automaton offline has exponential

---

[2] scopus.com

[3] ieeexplore.ieee.org

Table 1: Comparison of application-independent provably safe RL approaches.

Reference	**Verification Method**	**Space**		**Learning Tuple**	**Environment**
		State	Action		
Action replacement					
Akametalu et al. (2014)	HJI[1] reachability analysis	cont.	cont.	special RL alg.	1D quadrotor, cart-pole
Fisac et al. (2019)	HJI reachability analysis	cont.	cont.	N/A	1D quadrotor
Alshiekh et al. (2018)	model checking of automaton constructed from LTL[2]	disc.	disc.	*safe action*	Grid world
Könighofer et al. (2020)	model checking of automaton constructed from LTL	disc.	disc.	*adaption penalty*	ACC[3]
Anderson et al. (2020)	robust control invariant set	cont.	cont.	N/A	pendulum, reach-avoid, others
Hunt et al. (2021)	theorem proofing of $\mathsf{d}\mathcal{L}$[4] formulas	disc.	disc.	*naive*	Grid world
Bastani (2021)	MPC[5]	cont.	cont.	deployment only	bicycle, cart-pole
Shao et al. (2021)	Set-based reachability analysis	cont.	cont.	*naive*	3D quadrotor, highway driving
Selim et al. (2022b)	Set-based reachability analysis	cont.	cont.	*adaption penalty*	3D quadrotor, mobile robot
Action projection					
Pham et al. (2018)	verification of affine action constraints	cont.	cont.	*adaption penalty*	manipulator
Cheng et al. (2019)	CBF[6]	cont.	cont.	*safe action*	ACC, pendulum
Li et al. (2019a)	LTL and CBF	cont.	cont.	*adaption penalty*	manipulator
Gros et al. (2020)	MPC	cont.	cont.	*naive*	2D LTI[7] system
Wabersich & Zeilinger (2021)	MPC	cont.	cont.	*naive*	3D quadrotor, pendulum
Marvi & Kiumarsi (2022)	CBF	cont.	cont.	*adaption penalty*	2D LTI system
Selim et al. (2022a)	Set-based reachability analysis	cont.	cont.	*naive*	3D quadrotor, mobile robot
Kochdumper et al. (2023)	Set-based reachability analysis	cont.	cont.	*adaption penalty*	3D quadrotor
Action masking					
Fulton & Platzer (2018)	theorem proofing of $\mathsf{d}\mathcal{L}$ formulas	cont.	disc.	*naive*	ACC
Fulton & Platzer (2019)	theorem proofing of $\mathsf{d}\mathcal{L}$ formulas	cont.	disc.	*naive*	ACC
Huang & Ontañón (2022)	verification of affine action equality constraints	disc.	disc.	*naive*	Grid world
This study	Set-based reachability analysis	cont.	cont.	*naive*	pendulum, 2D quadrotor

Abbreviations: [1]Hamilton-Jacobi-Isaacs (HJI), [2]linear temporal logic (LTL), [3]adaptive cruise control (ACC), [4]differential dynamic logic ($\mathsf{d}\mathcal{L}$), [5]model predictive control (MPC), [6]control barrier function (CBF), [7]linear time-invariant (LTI).

complexity in state dimensions, and the online checking complexity also increases exponentially with the formula length (Baier & Katoen, 2008). In Alshiekh et al. (2018), the agent outputs n ranked actions, which are all checked for safety using $\varphi(\boldsymbol{s}, \boldsymbol{a})$. The shield then either executes the highest-ranked safe action or replaces the action with $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ if none of the n actions is safe. They update the agent with the *safe action* learning tuple but also propose that *both* can be used to obtain additional training information. In a similar fashion as Alshiekh et al. (2018), Könighofer et al. (2020) show that both probabilistic and deterministic shields increase the sample efficiency and performance for both action replacement and masking methods.

The authors in Akametalu et al. (2014) propose action replacement based on Hamilton-Jacobi-Isaacs reachability analysis, which was later extended by Fisac et al. (2019) to a general safe RL framework. They determine $\mathbb{S}_{\varphi}$using Hamilton-Jacobi-Isaacs reachability analysis given bounded system disturbances. The safety is then guaranteed by replacing any learned action on the border of $\mathbb{S}_{\varphi}$with $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ stemming from an Hamilton-Jacobi-Isaacs optimal controller to guide the system back inside the safe set. Hamilton-Jacobi-Isaacs reachability analysis can verify reach-avoid problems with arbitrary non-convex sets (Althoff, 2015) and disturbances that stem from a compact set. However, constructing the safe set offline scales exponentially in complexity with the number of states, which makes Hamilton-Jacobi-Isaacs reachability analysis infeasible for systems with more than four states (Chen & Tomlin, 2018). Fisac et al. (2019) argue that replacing the unsafe action with the action that maximizes the distance to the unsafe set increases performance

Table 2: Overview of applications in provably safe RL.

	Action Replacement	**Action Projection**	**Action Masking**
Aerial Vehicles	‡‡Akametalu et al. (2014); Shyamsundar et al. (2017); ‡‡Fisac et al. (2019); Anderson et al. (2020); †Harris & Schaub (2020); †Shao et al. (2021); †Selim et al. (2022b); †Nazmy et al. (2022)	†Wabersich & Zeilinger (2021); †Selim et al. (2022a); †Kochdumper et al. (2023)	N/A

Autonomous Driving	†Chen et al. (2020);
	Könighofer et al. (2020);
	†Shao et al. (2021);
	†Chen et al. (2022a);
	†Lee & Kwon (2022);
	‡‡Wang et al. (2023);
	†Evans et al. (2023)
		Cheng et al. (2019);
		†Wang (2022);
		†Hailemichael et al. (2022b);
		†Hailemichael et al. (2022a);
		‡‡Kochdumper et al. (2023)
			Fulton & Platzer (2018);
			†Mirchevska et al. (2018);
			Fulton & Platzer (2019);
			†Krasowski et al. (2020);
			Brosowsky et al. (2021);
			†Krasowski et al. (2022)

Power Systems	†Ceusters et al. (2023)	†Eichelbeck et al. (2022); †Chen et al. (2022b); †Zhang et al. (2023); †Yu et al. (2023)	†Tabas & Zhang (2022)
Robotic Manipulation	†Thumm & Althoff (2022)	‡‡Pham et al. (2018); ‡‡Li et al. (2019a)	N/A

Control Benchmarks	Akametalu et al. (2014);
	Anderson et al. (2020);
	Bastani (2021);
	Shao et al. (2021)
		Cheng et al. (2019);
		Gros et al. (2020);
		Wabersich & Zeilinger (2021);
		Marvi & Kiumarsi (2022)
			N/A

Grid World Games	Alshiekh et al. (2018); Hunt et al. (2021)	N/A	Huang & Ontañón (2022)

Miscellaneous	*Active suspension*:
	Li et al. (2019b);
	Computation:†Wang et al. (2022);
	Mobile robot: †Selim et al. (2022b)
		Mobile robot:
		†Selim et al. (2022a);
		Engine emission:
		†Norouzi et al. (2023)
			Computation:
			Seetanadi et al. (2020);
			Traffic signal:
			†Müller & Sabatelli (2022)

Note: Studies using high-fidelity simulators are marked with †, and ‡‡ indicates physical experiments. Additionally, papers occur multiple times in case they demonstrate their approach for different applications.

in uncertain real-world environments in comparison to action projection. However, Hamilton-Jacobi-Isaacs approaches suffer from the curse of dimensionality and are therefore only feasible for systems with specific characteristics (Herbert et al., 2021).

Shao et al. (2021) use a continuous reachability-based trajectory safeguard based on set-based reachability analysis for $\varphi(\boldsymbol{s}, \boldsymbol{a})$. Set-based reachability analysis is applicable to reach-avoid problems for general nonlinear systems with uncertainties in the initial state, system and input disturbances, and time delays as long as they stem from a compact set Althoff et al. (2021). The reachability analysis has polynomial offline and online complexity in state dimensions for most set representations as discussed by (Althoff et al., 2021). However, in comparison to Hamilton-Jacobi-Isaacs reachability analysis, set-based reachability analysis cannot handle arbitrary non-convex sets but depends on specific set representations. Shao et al. (2021) sample n new actions are sampled randomly in the vicinity of the agent's action if the agent's action is unsafe. They then execute the closest safe action to the original (unsafe) action. If none of the n new actions is safe, a failsafe action $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ is executed. Shao et al. (2021) train their agent on the *naive* learning tuple. Selim et al. (2022b) also verify the safety of actions with set-based reachability analysis. They propose an informed replacement for $\boldsymbol{\psi}(\boldsymbol{s})$ such that the reachable set of the controlled system are pushed away from the unsafe set $\mathbb{S} \setminus \mathbb{S}_{\varphi}$. The authors further propose a method to account for unknown system dynamics using a so-called black-box reachability analysis. They use the *adaption penalty* learning tuple and showcase that their approach achieves provable safety on three use cases.

Hunt et al. (2021) build the verification function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ using theorem proofing of differential dynamic logic formulas. Using $\varphi(\boldsymbol{s}, \boldsymbol{a})$, they determine $\mathbb{A}_{\varphi}(\boldsymbol{s})$ for discrete action spaces, and use $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ for replacement. They further show how provably safe end-to-end learning can be accomplished using controller and model monitors. They train the RL agent using the *naive* learning tuple on a drone example. The work of Anderson et al. (2020) proposes to define $\mathbb{S}_\varphi$ as a robust control invariant set and construct $\varphi(\boldsymbol{s}, \boldsymbol{a})$ from the single-step discrete system dynamics. A further notable work is Bastani (2021) that proposes a model predictive shield alongside the trained policy, which uses $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ for the replacement. Finally, Saunders et al. (2018) propose to use a human in the loop for the $\varphi(\boldsymbol{s}, \boldsymbol{a})$ and $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ functions in slow-paced environments.

The most popular method by publications is action replacement. This is also visible from the large variety of application-specific approaches, e.g., for aerial vehicles (Shyamsundar et al., 2017; Harris & Schaub, 2020; Nazmy et al., 2022), autonomous driving (Chen et al., 2020; 2022a; Lee & Kwon, 2022; Wang et al., 2023; Evans et al., 2023), power systems (Ceusters et al., 2023), robotic manipulation (Thumm & Althoff, 2022), active suspension system (Li et al., 2019b), and data traffic engineering for computation (Wang et al., 2022).

Action projection Research on action projection is usually conducted on continuous action and state spaces. The main differentiating factor between studies in this category is the specification of the optimization problem for the projection. To begin with, the work of Pham et al. (2018) guarantees safety using a differentiable constrained optimization layer, called OptLayer. Unfortunately, their approach is restricted to QP problems, so the system model and constraints have to be linear. Despite these limitations, they show the effectiveness of their approach on a collision avoidance task with a simple robotic manipulator.

Cheng et al. (2019) specify the safety constraints $\varphi(\boldsymbol{s}, \boldsymbol{a}) = 1$ of the optimization problem via CBFs. When CBF are given, this approach is advantageous as solving the optimization problem in (3) is polynomial in state dimensions. Theoretically, the CBF approach is applicable to general control-affine systems with disturbances from a compact set for reach-avoid specifications. However, guessing the CBF is not trivial and synthesizing them can be exponentially complex in system dimension (Ames et al., 2019). To solve (3) more efficiently online, Cheng et al. (2019) add a neural network to the approach in section 2.2, which approximates the correction of the CBF controllers on this action. The action is then shifted by the approximated value prior to optimization. This improves the implementation efficiency while still guaranteeing safety, as the action is often already safe after the shift and no optimization problem needs to be solved. Their safe learning with CBFs shows faster convergence speed in comparison to vanilla RL when learning on a pendulum and a car following task. Li et al. (2019a) propose a method to construct a continuous CBF from an automaton, which is defined by linear temporal logic formulas. They further construct a guiding reward from the given automaton to improve the learning performance. The proposed approach is capable of learning a high-dimensional cooperative manipulation task in a safe manner. The authors in Marvi & Kiumarsi (2022) define a different problem, where the system model is assumed to be deterministic but unknown. They learn an optimal controller and the system dynamics in an iterative fashion, while decreasing the conservativeness of their CBF in each iteration. The approach is provably safe for linear time-invariant (LTI) systems without disturbances.

Gros et al. (2020) and Wabersich & Zeilinger (2021) implement the optimization problem as a robust MPC problem as defined in (6). Despite its applicability to nonlinear systems, MPC are mostly used for linear systems when the controlled plant is fast in relation to the verification (Zeilinger et al., 2014). MPC is applicable to reach-avoid problems with measurement and state disturbances. Gros et al. (2020) mainly discuss how the learning update has to be adapted if action projection is used for different RL algorithms. For Q-learning, they find that no adaption of the learning algorithm is necessary when the *naive* tuple is used. For policy gradient methods, they argue that the projection must also be included in the gradient for stable learning (Gros et al., 2020, Sec. 3). One downside of the robust MPC formulation of Gros et al. (2020) and Wabersich & Zeilinger (2021) is that dynamic constraints originating from moving obstacles or persons in the environment are not trivial to integrate. They approximate the dynamics of the system with a Gaussian process (GP), so that absolute safety is not given. However, they could guarantee absolute safety if they would assume bounded dynamics of the environment as the aforementioned approaches do. Gros et al. (2020) evaluate their approach on a simple 2D LTI system and Wabersich & Zeilinger (2021) show the efficacy of their approach on a pendulum and a quadrotor task.

Contrary to their previous work in Selim et al. (2022b), Selim et al. (2022a) propose to solve an optimization problem to find the closest safe action instead of using an informed replacement. They again use set-based reachability analysis to construct $\varphi(\boldsymbol{s}, \boldsymbol{a})$. They test their approach on a quadrotor and mobile robot benchmark. Kochdumper et al. (2023) utilize set-based reachability analysis to verify actions in $\varphi(\boldsymbol{s}, \boldsymbol{a})$. They formulate the projection for the generator's parameterization of the action space zonotope and arrive at a mixed-integer quadratic problem with polynomial constraints. Their approach achieves provable safety for non-linear systems with bounded disturbances and they demonstrate their approach on two quadrotor tasks, autonomous driving on highways and on a physical f1tenth car.

Next to the conceptual approaches, action projection algorithms are also specifically proposed for many cyber-physical systems such as autonomous driving (Wang, 2022; Hailemichael et al., 2022a;b), power systems (Eichelbeck et al., 2022; Chen et al., 2022b; Zhang et al., 2023; Yu et al., 2023), and engine emission control (Norouzi et al., 2023).

Action masking To the best of our knowledge, all existing literature considers action masking only for discrete action spaces. The work Huang & Ontañón (2022) analyzes the effect of discrete action masking on the policy gradient algorithm in RL, but they assume that $\mathbb{A}_s$ is known, which is only the case in game and grid world environments.

The two main works investigating action masking are Fulton & Platzer (2018) and Fulton & Platzer (2019). They construct controller and model monitors based on theorem proofing of differential dynamic logic specifications, which is introduced by Platzer (2008). The controller monitor is used to build the mask $\boldsymbol{\eta}(\boldsymbol{s})$. The model monitor verifies if the underlying system model is correct based on previous transitions. In each state, the agent can choose from the set of actions that the controller monitor verified as safe. To identify the correct system can be challenging, thus an approach to automatically generate candidates is introduced as well. Their approach is provably safe if the initial model is correct (Fulton & Platzer, 2018) or multiple models are given, from which at least one is correct (Fulton & Platzer, 2019) for all times. They validate their provably safe action masking on adaptive cruise control tasks. Additionally, we want to highlight that our (this) work is the first to introduce action masking for continuous action spaces as described in section 2.3, and is therefore included in Table 1.

In addition to the aforementioned works, there are works investigating action masking for the specific application of autonomous driving (Mirchevska et al., 2018; Krasowski et al., 2020; Brosowsky et al., 2021; Krasowski et al., 2022), power systems (Tabas & Zhang, 2022), adaptive routing for computation (Seetanadi et al., 2020), and urban traffic signal control (Müller & Sabatelli, 2022).

4 Experimental Comparison

In this section, we evaluate the performance of the three provably safe RL classes and the four learning tuples introduced in section 2. For our comparison, we select an inverted pendulum and a 2D quadrotor stabilization task, as they are benchmarks commonly evaluated in related works presented in table 1, the provably safe state set $\mathbb{S}_\varphi$ is the same and therefore comparable for all three approaches, and the computation time allows for exhaustive comparison of different RL algorithms and learning tuples. We add system noise to the benchmarks to make them more realistic and show that the provably safe RL approaches can handle noise sampled from a compact noise set. The added noise makes our experiments more realistic than most evaluations in table 1. Despite their low-dimensionality, our results are likely transferable to real-world systems, since real-world complexity is often reduced in practice by using lower-dimensional abstract models and an additional noise term. Conformance checking techniques (Roehm et al., 2019; Liu et al., 2023) can then guarantee that the abstract model incorporates the real-world behavior of the system.

The algorithms shown in this section are action replacement with $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$, action projection using the CBFs implementation, and action masking. We compare each configuration on ten random seeds and five common RL implementations[4] continuous Twin Delayed Deep Deterministic policy gradient algorithm (TD3) (Fujimoto et al., 2018), continuous soft actor-critic (SAC) (Haarnoja et al., 2018), discrete DQN (Mnih et al., 2013), and continuous and discrete proximal policy optimization (PPO) (Schulman et al., 2017).

---

[4] All implementations are based on `stable-baselines3` (Raffin et al., 2021).

4.1 Environments

We compare the provably safe RL approaches on an inverted pendulum and a 2D quadrotor stabilization task.

Inverted Pendulum The state of the pendulum is defined as $\boldsymbol{s} = \left[\theta, \dot{\theta}\right]$, and follows the dynamics

$$\dot{\boldsymbol{s}} = \begin{pmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) + \frac{1}{ml^2}a \end{pmatrix}, \tag{12}$$

where a is the one-dimensional action, g is gravity, m is the mass of the pendulum, l its length, and friction and damping are ignored. We discretize the dynamics using the Euler method. The input lies within $[a_{\min}, a_{\max}] = [-30\text{rad}\,\text{s}^{-1}, 30\text{rad}\,\text{s}^{-1}]$. The desired equilibrium state is $\boldsymbol{s}^* = [0, 0]$. The observation and reward are the same as for the *OpenAI Gym Pendulum-V0*^{5} environment.

2D quadrotor The quadrotor can only fly in the vertical x-z-plane and rotate around the y-axis with angle θ. The state of the system is defined as $\boldsymbol{s} = \left[x, z, \dot{x}, \dot{z}, \theta, \dot{\theta}\right]$ and the action as $\boldsymbol{a} = [u_1, u_2]$. The system dynamics are

$$\dot{\boldsymbol{s}} = \begin{pmatrix} \dot{x} \\ \dot{z} \\ u_1 k \sin(\theta) + w_1 \\ -g + u_1 k \cos(\theta) + w_2 \\ \dot{\theta} \\ -d_0\theta - d_1\dot{\theta} + n_0 u_2 \end{pmatrix} \tag{13}$$

where the noise is sampled uniformly from a compact noise set $\boldsymbol{w} = [w_1, w_2] \sim \mathbb{W} \subset \mathbb{R}^2$, and k, d_0, d_1, and n_0 are constant system parameters (see Table 4). We linearize the dynamics with a first-order Taylor expansion at the equilibrium point $\boldsymbol{s}^* = [0, 1, 0, 0, 0, 0]$ and solve the differential equations for a time-discrete system. The input ranges from $\boldsymbol{a}_{\min} = [-1.5 + \frac{g}{K}, -\frac{\pi}{12}]$ to $\boldsymbol{a}_{\max} = [1.5 + \frac{g}{K}, \frac{\pi}{12}]$. The reward is defined as $r = \exp\left(-\left\|\boldsymbol{s} - \boldsymbol{s}^*\right\| - 0.01\sum_{i=1}^{2}\frac{a_t[i] - a_{\min}[i]}{a_{\max}[i] - a_{\min}[i]}\right)$.

4.2 Computation of the Safe State Set

To obtain a possibly large set of provably safe states $\mathbb{S}_\varphi$ and a failsafe controller for our environments, we use the scalable approach for computing robust control invariant sets presented in Schäfer et al. (2023): for every state $\boldsymbol{s}_t \in \mathbb{S}_\varphi$, there exists a failsafe action $\tilde{\boldsymbol{a}}_t \in \mathbb{A}$ so that $\boldsymbol{s}_{t+1} = \boldsymbol{g}(\boldsymbol{s}_t, \tilde{\boldsymbol{a}}_t) \in \mathbb{S}_\varphi$. Since $\mathbb{S}_\varphi \subseteq \mathbb{S}$, satisfaction of our safety specifications $\boldsymbol{s} \in \mathbb{S}$ and $\tilde{\boldsymbol{a}} \in \mathbb{A}$ for every future point in time follows by induction. Using the algorithm in Schäfer et al. (2023) provides us with an explicit representation of $\mathbb{S}_\varphi$ with maximized volume, which enables a fair comparison of our provably safe RL implementations.

To retrieve $\mathbb{A}_\varphi$ from $\mathbb{S}_\varphi$ at a given state $\boldsymbol{s}$, we first convert $\mathbb{S}_\varphi$ from a so-called generator representation used in Schäfer et al. (2023) into half-space representation, i.e., $\mathbb{S}_\varphi = \{\boldsymbol{s} | \boldsymbol{C}\boldsymbol{s} \leq \boldsymbol{d}\}$ using the open-source toolbox CORA (Althoff, 2015). We then formulate the dynamics of the system in the form of (4) with an additional noise term $\boldsymbol{g}(\boldsymbol{s}, \boldsymbol{a}) = \boldsymbol{g}_a(\boldsymbol{s})\boldsymbol{a} + \boldsymbol{g}_u(\boldsymbol{s}) + \mathrm{w}$. The noise w is assumed to stem from a compact set $\boldsymbol{w} \sim \mathbb{W}$. Finally, we reformulate the constraints as

$$\boldsymbol{C}\boldsymbol{g}_a(\boldsymbol{s})\boldsymbol{a} \leq \boldsymbol{d} - \boldsymbol{C}\boldsymbol{g}_u(\boldsymbol{s}) - \boldsymbol{C}\mathrm{w}. \tag{14}$$

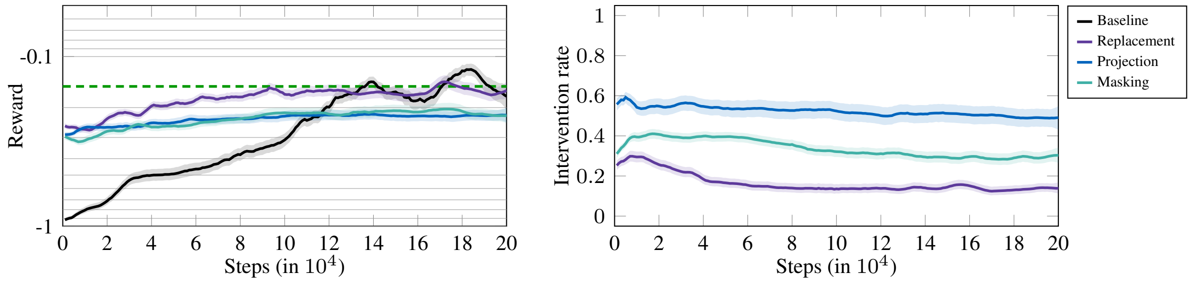
This results in a polytope with $\dim(\boldsymbol{d}) \times 2^{\dim(\mathbb{W})}$ halfspaces. Finally, we use `pypoman`^{6} to obtain the halfspace-representation of $\mathbb{A}_\varphi$.

In theory, the approach of Schäfer et al. (2023) allows us to calculate $\mathbb{S}_\varphi$ for nonlinear systems with up to 20 dimensions, such as chemical reactors. However, the conversion to halfspace representation is computational too complex for higher dimensional systems. Therefore, we plan to develop generator-based versions of our provably safe RL methods in future work to mitigate these shortcomings.

---

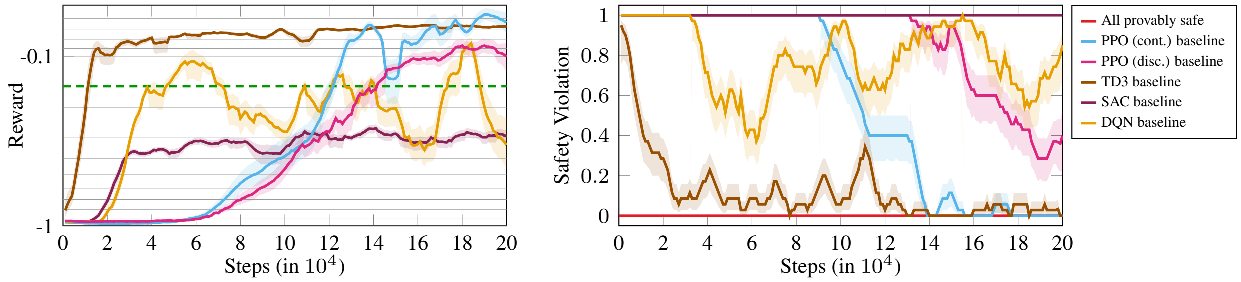
[5] `gymnasium.farama.org/environments/classic_control/pendulum/`

[6] Available at https://github.com/stephane-caron/pypoman



(a) Reward provably safe RL algorithms

(b) Intervention rate provably safe RL algorithms



(c) Reward baselines

(d) Safety violation rate baselines

Figure 2: Training curves for the 2D quadrotor benchmark. Top: Comparison of benchmark algorithms TD3, SAC, DQN, PPO continuous and discrete on ten random seeds per algorithm. Bottom: Comparison of the three provably safe RL classes and the unsafe benchmarks averaged over all algorithms trained with the *naive* tuple. The left column depicts the reward. The right column shows the safety violations for the baselines, and the safety intervention rate for the provably safe RL algorithms.

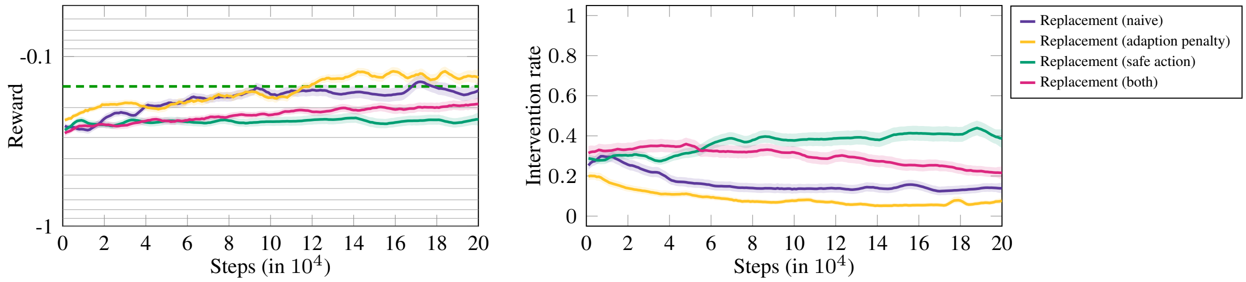
4.3 Results

The 2D quadrotor task functions as main comparison environment for this work as it is more complex and shows the differences between provably safe RL approaches clearer than the inverted pendulum task. We evaluate the differences between the provably safe RL algorithms in figure 2 and the effect of different learning tuples in figure 3. All training runs on all individual algorithms and both environments, including a comparison between the $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ and $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ replacement function, are displayed in the Appendix.

Comparison of provably safe RL algorithms The safety violation evaluation of the baselines in figure 2d shows that the baseline algorithms fail to guarantee safety during training in the 2D quadrotor stabilization task. All provably safe RL algorithms guarantee safety as expected following from the conceptual analysis in section 2. Between the baselines, TD3 converges significantly faster than all other algorithms.

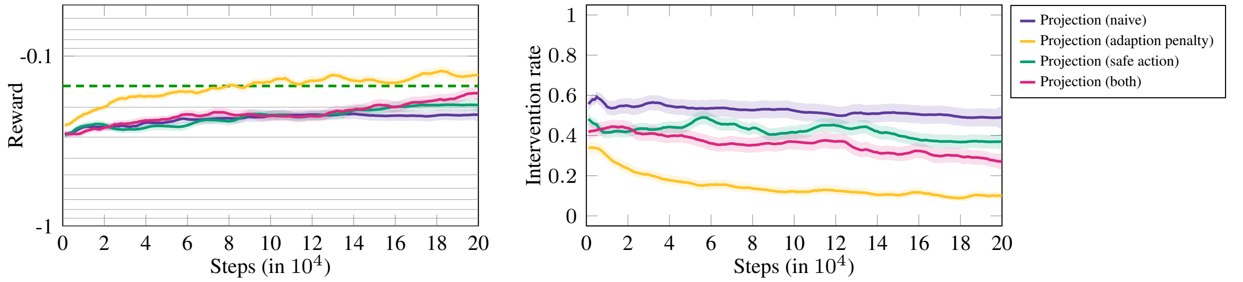
Figures 2a and 2b show the performance of the three provably safe RL categories action replacement using $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$, action projection, and action masking together with the baselines averaged over all five RL implementations and trained using the *naive* learning tuple. For better comparison of the reward curves in figure 2a, we added a dashed green line that indicates the final training reward averaged over all five RL baselines and ten random seeds. The reward comparison shows that action replacement performs better than action projection and masking on average.

To compare how active the provably safe RL methods are, we introduce the intervention rate metric. For action replacement and projection, this indicates the average rate of RL steps in which the action was altered by the safety function. For action masking, the intervention rate compares the average volume of the provably safe action set over an episode with the volume of the provably safe action set at the equilibrium point of the system, e.g., $V_{\mathbb{A}_\varphi,\text{episode}}/V_{\mathbb{A}_\varphi,\text{equi}}$. Figure 2b shows that action replacement relies significantly



(a) Reward action replacement

(b) Intervention rate action replacement



(c) Reward action projection

(d) Intervention rate action projection

Figure 3: Evaluation of the training tuples for the 2D quadrotor averaged over the five RL algorithms TD3, SAC, DQN, PPO continuous and discrete with ten random seeds each. The left column depicts the reward and the right column the safety intervention rate. The top row shows the different learning tuples for action replacement and the bottom row for action projection.

less on the safety mechanism than projection and masking. In general, we report that a lower intervention rate often coincides with a higher reward.

Comparison of learning tuples We evaluate the impact of different learning tuples on the performance and safety activity averaged over all five RL algorithms in figure 3. When action masking is used, only safe actions can be sampled, i.e., only the *naive* tuple is meaningful; so we omit action masking from this evaluation. For both action replacement and projection the *adaption penalty* tuple leads to the highest performance and lowest safety intervention rate, even outperforming the average over the baselines. In action projection, the *naive* tuple performs significantly worse than in action replacement. The *safe action* and *both* tuple seem to be only beneficial when using action projection and decrease performance when using action replacement.

5 Discussion

Our experiments confirm the theoretical statement that provably safe RL methods are always safe when assumption 1 is fulfilled. In the two tested environments, the investigated RL baselines show non-zero safety violations during training, even after convergence. Subsequently, we discuss five statements resulting from the experiments, provide intuitions for implementing provably safe RL, summarize the limitations, and identify future research directions in provably safe RL.

Selecting a provably safe RL approach First, we want to summarize our experience with the three provably safe RL classes to give the reader an intuition when to choose which method. Action replacement is the easiest method to implement for continuous action spaces. It shows very good performance and low intervention rates. Hereby, using a failsafe action for replacement with an adaption penalty is both simple to implement and usually among the best performing methods in our experiments, and is thereby recommended if available. Still, the random sampling safe actions might outperform the failsafe action. So if sampling from the safe action space is easily available, e.g., in discrete action spaces, a failsafe controller can be omitted. Action projection tends to be problematic in practice due to floating point errors and other small numerical errors, resulting in infeasible optimization problems. We, therefore, have to use methods like the provably safe MPC in Schürmann et al. (2018), or use a failsafe controller if the optimization problem is not solvable. Together with the higher intervention rates compared to action replacement and the complex implementation, we would usually not recommend to use action projection. However, if one already has a CBF or MPC formulation, it might be the most suitable solution. Action masking is particularly easy to implement for discrete action spaces and shows good performance in that setting. However, action masking can be difficult to implement or very restrictive for continuous action spaces, where the safe action space significantly diverges from an axis-aligned box. Generally, the different approaches can also be combined to some extend. For example, if the optimization problem for action projection becomes infeasible, a failsafe controller can be used.

Selecting a learning tuple The *adaption penalty* learning tuple performed best, especially when using action projection. In our experiments, a simple constant reward penalty already improved the performance. Other environments may require more careful reward tuning, or the *adaption penalty* tuple could fail altogether due to reward hacking (Skalse et al., 2022) or goal misgeneralization (Langosco et al., 2022). The results in figure 3 further show that using the safe action $\tilde{\boldsymbol{a}}$ in the training tuple, i.e., configurations *both* and *safe action*, benefit the performance of action projection methods but impair the training with action replacement. This effect can result from the fact that action replacement alters the action more than action projection, leading to a lower likelihood that the altered action stems from the RL policy. The evaluations in the Appendix show that this effect is prominent when using PPO. This on-policy algorithm assumes that the current batch of training data stems from the current policy. Hence, we would recommend using the *adaption penalty* tuple when possible and only using *safe action* in combination with action projection.

Convergence The provably safe RL training converges similarly fast or faster than the baselines, while the performance (measured by the reward) at the beginning of the training is better for provably safe RL agents. One reason for the convergence difference is the different exploration setting. On the one hand, the baseline agents need to explore more actions including actions that do not lead to reaching the goal. On the other hand, safe agents do not get as much information about the environment as only safe states are explored. Another reason for differences in convergence can be that the provably safe RL approaches potentially correct the action selected by the agent. Since this correction for action replacement and action projection happens after the forward pass through the policy, the gradient calculation might need correction as well. So far, there is only little theoretic work on this (Hunt et al., 2021; Gros et al., 2020) and it is unclear if in practice a correction of the gradient is necessary or using the *adaption penalty* tuple is sufficient. Generally, it should be mentioned that the safety verification creates a computational overhead such that convergence speed is not necessarily aligned with the elapsed real time.

Computational complexity The computational complexity of the three approaches highly depends on the scenario-specific implementation. For action projection, the main implementation challenge is to guarantee that the optimization problem is always feasible. If the optimization problem can be formulated as a QP problem, then the computational complexity is polynomial as shown by Vavasis (2001). Contrary, the computational complexity of action replacement and action masking highly depends on the algorithm that identifies the safety of actions. For discrete action masking, the computational complexity scales linearly with the total number of actions $\mathcal{O}(|\mathbb{A}|)$. While for action replacement, we only need a single safe action, so in the ideal case, e.g., using a failsafe controller, the computational complexity is constant in regard to the total number of actions $\mathcal{O}(1)$. If the action replacement approach needs to determine the entire set of safe actions, they have the same computational complexity as action masking. The computational complexity for identifying the continuous safe action space depends on the task-specific implementation. One possibility is to compute the safe action space using set-based reachability analysis, where we want to point the interested reader to Althoff (2015) for different implementations.

Online vs. offline implementation Online and offline have two notions in provably safe RL: online vs. offline safety verification and online vs. offline RL. The safety function usually needs to be evaluated online since for continuous state spaces pre-computing the safe action set for all state is not feasible. Thus, the computational complexity of the safety function is important for real-time applications as discussed in the previous paragraph. If the state and action space are discrete, it can be possible to pre-compute the safe actions offline Alshiekh et al. (2018); Huang & Ontañón (2022). Generally, only if the safety function is integrated in RL between the agent and environment to correct actions (see figure 1), safety is guaranteed. In this study, we focus on comparing online and off-policy RL algorithms Levine et al. (2020) since they are used for existing provably safe RL research. Still, provably safe RL can also be used for offline RL while the safety function would be integrated during deployment and most likely also during the data gathering phase if this phase is conducted in a safety-critical environment. However, more specific advice on offline provably safe RL needs to be substantiated with experimental evaluations and, thus, is a topic for future research.

Limitations of provably safe RL There are limitations for provably safe RL that follow from the conceptual analysis in section 2. Most importantly, safety has to be decidable, i.e., there must be a safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$, which fulfills assumption 1. For this safety function, there is system knowledge necessary, and especially for systems with a high number of continuous state variables, the safety function is potentially hard to retrieve. Additionally, safety guarantees are strongly tied to the safety function. If the safety function provides complex guarantees (e.g., temporal logic), it is usually computationally more expensive than for simpler guarantees (e.g., system stays within safe state set). Second, safety can only be decided if the state of the system is correctly observed within noise bounds. Thus, for a fully provably safe system, the perception module also needs to be verified such that it provides observations that are correct within the noise bounds. Third, there is often a trade-off between safety and performance since for many tasks these two objectives are only partially aligned. For example, if a car drives faster it reaches its destination earlier but collisions are more difficult to avoid with higher speed. Since provably safe RL only allows safe behaviour, there is no such trade-off as safety is always prioritized over performance. Thus, the safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ cannot be too conservative since then the agent would only perform actions that are safe for an infinite time horizon, e.g., often the stand still action. Lastly, comparing provably safe RL approaches is challenging as we need to define a safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ that is efficiently usable by different approaches. Furthermore, the notion of safety is usually application-specific, so different application-specific approaches are hard to compare. We provide the first comparison of provably safe RL on two common continuous stabilization tasks, but further research is necessary to make stronger claims about the best provably safe RL approach.

Future research based on proposed taxonomy Most action projection approaches discussed in section 3 project the RL action on the edge of $\mathbb{A}_\varphi$. In our experiments, we encountered two negative side effects related to this action projection implementation: First, the projection to the edge of $\mathbb{A}_\varphi$ often leads to a relatively small $\mathbb{A}_\varphi$ in the next RL step, quickly resulting in a very small set $\mathbb{A}_\varphi$ after a few unsafe RL actions. Second, small floating point errors can cause unsafe actions and must be handled. Therefore, future action projection research should investigate the objective function of (3) to obtain more stable training. Action masking is a promising technique but has mainly been used in grid world environments and games, e.g., the Atari benchmark Huang & Ontañón (2022), with discrete action spaces so far. Our proposed continuous action masking approach only applies to specific environments and performed well for the pendulum but showed mixed results for the 2D quadrotor. Thus, future research should investigate ways to extend continuous action masking to general convex or even non-convex $\mathbb{A}_\varphi$ representations in order to improve its applicability to more complex benchmarks and reduce the conservativeness of $\mathbb{A}_\varphi$. Additionally, it should be investigated if the agent should be informed about the reduction of the action space through masking. This could result in improved convergence and an agent that is less dependent on the safety verification, similar to the effect of the *adaption penalty* tuple for action replacement and action projection. The evaluation of the considered benchmark shows that action replacement performs better than action projection and masking as discussed previously. However, it is still unclear if the replacement strategy $\boldsymbol{\psi}(\boldsymbol{s})$ is important for the convergence and performance of the agent, especially when applied to more complex tasks. Thus, future action replacement research should empirically and theoretically investigate this question.

Improving applicability of provably safe RL Despite the promising previous work discussed in section 3, there are few works on high-dimensional non-linear systems and limited real-world applications. We suggest five major factors where future research would improve applicability. First, some approaches need to be computationally more efficient to be real-world applicable. Especially the computational efficiency of verification methods is relevant and should be improved as discussed previously. Second, we observe that the type of learning tuple used has a significant influence on the performance of the agent for some RL algorithms. Also, there is little theoretic research on how provably safe RL approaches influence convergence to an optimal policy. More empirical and theoretical research on the convergence effects of provably safe RL and its learning tuples would support effective and efficient implementation of the algorithms. Third, to evaluate new provably safe RL approaches common benchmarks are necessary. Additionally, the three action correction strategies should be compared on more complex benchmarks to clarify if our observations can be extended to them. Such benchmarking would make the provably safe RL research more comparable, would ease starting research on provably safe RL, and would provide more evidence to decide for the best-suited provably safe RL approach. Fourth, recent work shows a low variety of safety specifications, mainly comprising stabilization and reach-avoid specifications. Contrary, real-world safety is more complex, e.g., traffic rules such as waiting at a red light and safely but quickly moving at a green light. Finally, provably safe RL requires expert knowledge on verification methods. Future research could mitigate this through modular and automatic approaches, where fewer engineering decisions are necessary, and more parameters can be tuned automatically. With these advances, provably safe RL could bring the best of RL and formal specifications together towards RL methods that require as little expert knowledge as necessary and provide formal guarantees for complex safety specifications to achieve reliable and trustworthy cyber-physical systems.

6 Conclusion

In conclusion, we define a categorization for provably safe RL methods that structures the literature from a learning perspective. We present these provably safe RL methods from a conceptual perspective and define necessary assumptions. Our proposed categorization into action replacement, action projection, and action masking supports researchers in comparing their works and provide valuable insights into the selection process of provably safe RL methods. The comparison of four implementations of provably safe RL on a 2D quadrotor and an inverted pendulum stabilization benchmark provides further insights on the best-suited method for different tasks. We further give practical recommendations for selecting a provably safe RL approach and learning tuple, which will be a valuable addition for researchers that are new to RL or formal methods. Lastly as discussed in section 5, our proposed taxonomy and experimental evaluation yield multiple promising future research directions.

Acknowledgments

Anonymous for double-blind peer review.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 22–31, 2017.

Anayo K. Akametalu, Shahab Kaynama, Jaime F. Fisac, Melanie N. Zeilinger, Jeremy H. Gillula, and Claire J. Tomlin. Reachability-based safe learning with Gaussian processes. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 1424–1431, 2014.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pp. 2669–2678, 2018.

Matthias Althoff. An Introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pp. 120–151, 2015.

Matthias Althoff, Goran Frehse, and Antoine Girard. Set Propagation Techniques for Reachability Analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):369–395, 2021.

Eitan Altman. Constrained Markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical Methods of Operations Research*, 48(3):387–417, 1998.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control Barrier Functions: Theory and Applications. In *European Control Conference (ECC)*, pp. 3420–3431, 2019.

Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic Reinforcement Learning with Formally Verified Exploration. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, volume 33, pp. 6172–6183, 2020.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press, 2008.

Osbert Bastani. Safe Reinforcement Learning with Nonlinear Dynamics via Model Predictive Shielding. In *Proc. of the American Control Conf. (ACC)*, pp. 3488–3494, 2021.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable Reinforcement Learning via Policy Extraction. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, pp. 2499–2509, 2018.

Felix Berkenkamp, Angela P Schoellig, Matteo Turchetta, and Andreas Krause. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, pp. 908–918, 2017.

Sara Bouraine, Thierry Fraichard, and Hassen Salhi. Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 174–179, 2012.

Mathis Brosowsky, Florian Keck, Jakob Ketterer, Simon Isele, Daniel Slieter, and Marius Zöllner. Safe Deep Reinforcement Learning for Adaptive Cruise Control by Imposing State-Specific Safe Sets. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 488–495, 2021.

Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Glenn Ceusters, Luis Ramirez Camargo, Rüdiger Franke, Ann Nowé, and Maarten Messagie. Safe reinforcement learning for multi-energy management systems with known constraint functions. *Energy and AI*, 12, 2023.

Dong Chen, Longsheng Jiang, Yue Wang, and Zhaojian Li. Autonomous Driving using Safe Reinforcement Learning by Incorporating a Regret-based Human Lane-Changing Decision Model. In *Proc. of the American Control Conf. (ACC)*, pp. 4355–4361, 2020.

Mo Chen and Claire J Tomlin. Hamilton–Jacobi Reachability: Some Recent Theoretical Advances and Applications in Unmanned Airspace Management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):333–358, 2018.

Shengduo Chen, Yaowei Sun, Dachuan Li, Qiang Wang, Qi Hao, and Joseph Sifakis. Runtime Safety Assurance for Learning-enabled Control of Autonomous Driving Vehicles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 8978–8984, 2022a.

Yize Chen, Yuanyuan Shi, Daniel Arnold, and Sean Peisert. SAVER: Safe Learning-Based Controller for Real-Time Voltage Regulation. In *Proc. of the IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, 2022b.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pp. 3387–3395, 2019.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-Based Approach to Safe Reinforcement Learning. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, pp. 8103–8112, 2018.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, 2017.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv*, abs/1801.0, 2018.

Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 128–136, 2021.

Michael Eichelbeck, Hannah Markgraf, and Matthias Althoff. Contingency-constrained economic dispatch with safe reinforcement learning. In *Proc. of the IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, pp. 597–602, 2022.

Mohamed El-Shamouty, Xinyang Wu, Shanqi Yang, Marcel Albus, and Marco F. Huber. Towards Safe Human-Robot Collaboration Using Deep Reinforcement Learning. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, pp. 4899–4905, 2020.

Benjamin D Evans, Hendrik W Jordaan, and Herman A Engelbrecht. Safe reinforcement learning for high-speed autonomous racing. *Cognitive Robotics*, 3:107 – 126, 2023.

Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2019.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 2587–2601, 2018.

Nathan Fulton and André Platzer. Safe Reinforcement Learning via Formal Methods: Toward Safe Control Through Proof and Learning. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, pp. 6485–6492, 2018.

Nathan Fulton and André Platzer. Verifiably Safe Off-Model Reinforcement Learning. In *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 413–430, 2019.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.

Jeremy H. Gillula and Claire J. Tomlin. Reducing Conservativeness in Safety Guarantees by Learning Disturbances Online: Iterated Guaranteed Safe Online Learning. *Robotics: Science and Systems*, 8(1): 81–88, 2013.

Sebastien Gros, Mario Zanon, and Alberto Bemporad. Safe Reinforcement Learning via Projection on a Safe Set: How to Achieve Optimality? *IFAC-PapersOnLine*, 53(2):8076–8081, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 1861–1870, 2018.

Habtamu Hailemichael, Beshah Ayalew, Lindsey Kerbel, Andrej Ivanco, and Keith Loiselle. Safety Filtering for Reinforcement Learning-based Adaptive Cruise Control. *IFAC-PapersOnLine*, 55(24):149–154, 2022a.

Habtamu Hailemichael, Beshah Ayalew, Lindsey Kerbel, Andrej Ivanco, and Keith Loiselle. Safe Reinforcement Learning for an Energy-Efficient Driver Assistance System. In *IFAC-PapersOnLine*, volume 55:37, pp. 615 – 620, 2022b.

Andrew Harris and Hanspeter Schaub. Spacecraft command and control with safety guarantees using shielded deep reinforcement learning. In *AIAA Scitech 2020 Forum*, volume 1, 2020.

Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J. Pappas, and Insup Lee. Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 5338–5343, 2019a.

Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Towards verifiable and safe model-free reinforcement learning. In *Proc. of the Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, pp. 1–9, 2019b.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. In *Proc. of the Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 483–491, 2020.

Matthias Heger. Consideration of Risk in Reinforcement Learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 105–111, 1994.

Sylvia Herbert, Jason J. Choi, Suvansh Sanjeev, Marsalis Gibson, Koushil Sreenath, and Claire J. Tomlin. Scalable Learning of Safety Guarantees for Autonomous Systems using Hamilton-Jacobi Reachability. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 5914–5920, 2021.

Lukas Hewing, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Shengyi Huang and Santiago Ontañón. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *The Int. Florida Artificial Intelligence Research Society Conf. Proc. (FLAIRS)*, 35, 2022.

Nathan Hunt, Nathan Fulton, Sara Magliacane, Trong Nghia Hoang, Subhro Das, and Armando Solar-Lezama. Verifiably safe exploration for end-to-end reinforcement learning. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pp. 1–11, 2021.

B. Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A.Al Sallab, Senthil Yogamani, and Patrick Perez. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2021.

Niklas Kochdumper, Hanna Krasowski, Xiao Wang, Stanley Bak, and Matthias Althoff. Provably Safe Reinforcement Learning via Action Projection Using Reachability Analysis and Polynomial Zonotopes. *IEEE Open Journal of Control Systems*, 2:79–92, 2023.

Bettina Könighofer, Florian Lorber, Nils Jansen, and Roderick Bloem. Shield Synthesis for Reinforcement Learning. In *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, pp. 290–306, 2020.

Bettina Könighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler, and Roderick Bloem. Online Shielding for Stochastic Systems. In *Proc. of the NASA Formal Methods (NFM)*, pp. 231–248, 2021.

Hanna Krasowski, Xiao Wang, and Matthias Althoff. Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction. In *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, pp. 1–7, 2020.

Hanna Krasowski, Yinqiang Zhang, and Matthias Althoff. Safe Reinforcement Learning for Urban Driving using Invariably Safe Braking Sets. In *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, pp. 2407–2414, 2022.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal Misgeneralization in Deep Reinforcement Learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 12004–12019. PMLR, 2022.

Dongsu Lee and Minhae Kwon. ADAS-RL: Safety learning approach for stable autonomous driving. *ICT Express*, 8(3):479 – 483, 2022.

Sergey Levine, Kumar Aviral, Tucker George, and Fu Justin. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv*, (2005.01643), 2020.

Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37), 2019a.

Zhaojian Li, Tianshu Chu, and Uroš Kalabić. Dynamics-Enabled Safe Deep Reinforcement Learning: Case Study on Active Suspension Control. In *Proc. of the IEEE Conf. on Control Technology and Applications (CCTA)*, pp. 585–591, 2019b.

Zemin Eitan Liu, Quan Zhou, Yanfei Li, Shijin Shuai, and Hongming Xu. Safe Deep Reinforcement Learning-based Constrained Optimal Control Scheme for HEV Energy Management. *IEEE Transactions on Transportation Electrification*, pp. 1f, 2023.

Tommaso Mannucci, Erik-Jan van Kampen, Cornelis de Visser, and Qiping Chu. Safe Exploration Algorithms for Reinforcement Learning Controllers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(4):1069–1081, 2018.

Zahra Marvi and Bahare Kiumarsi. Safe reinforcement learning: A control barrier function optimization approach. *Int. Journal of Robust and Nonlinear Control*, 31(6):1923–1940, 2021.

Zahra Marvi and Bahare Kiumarsi. Reinforcement Learning With Safety and Stability Guarantees During Exploration For Linear Systems. *IEEE Open Journal of Control Systems*, 1:322–334, 2022.

Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker. High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning. In *Proc. of the Int. Conf. on Intelligent Transportation Systems (ITSC)*, pp. 2156–2162, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv*, abs/1312.5, 2013.

Arthur Müller and Matthia Sabatelli. Safe and Psychologically Pleasant Traffic Signal Control with Reinforcement Learning using Action Masking. *Proc. of the IEEE Conf. on Intelligent Transportation Systems (ITSC)*, pp. 951–958, 2022.

Islam Nazmy, Andrew Harris, Morteza Lahijanian, and Hanspeter Schaub. Shielded Deep Reinforcement Learning for Multi-Sensor Spacecraft Imaging. In *Proc. of the American Control Conf. (ACC)*, pp. 1808–1813, 2022.

Armin Norouzi, Saeid Shahpouri, David Gordon, Mahdi Shahbakhti, and Charles Robert Koch. Safe deep reinforcement learning in diesel engine emission control. *Proc. of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 2023.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing Generalization in Deep Reinforcement Learning, 2019.

Christian Pek, Stefanie Manzinger, Markus Koschi, and Matthias Althoff. Using online verification to prevent autonomous vehicles from causing accidents. *Nature Machine Intelligence*, 2(9):518–528, 2020.

Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. OptLayer - Practical Constrained Optimization for Deep Reinforcement Learning in the Real World. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 6236–6243, 2018.

André Platzer. Differential Dynamic Logic for Hybrid Systems. *Journal of Automated Reasoning*, 41(2): 143–189, 2008.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Hendrik Roehm, Jens Oehlerking, Matthias Woehrle, and Matthias Althoff. Model conformance for cyber-physical systems: A survey. *ACM Transactions on Cyber-Physical Systems*, 3(3):1–26, 2019.

William Saunders, Girish Sastry, Andreas Stuhlmüller, and Owain Evans. Trial without Error: Towards Safe Reinforcement Learning via Human Intervention. In *Proc. of the Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 2067 – 2069, 2018.

Lukas Schäfer, Felix Gruber, and Matthias Althoff. Scalable Computation of Robust Control Invariant Sets of Nonlinear Systems. *IEEE Transactions on Automatic Control*, pp. 1–15, 2023.

Lukas M Schmidt, Georgios D Kontes, Axel Plinge, and Christopher Mutschler. Can you trust your autonomous car? Interpretable and verifiably safe reinforcement learning. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 171–178, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv*, abs/1707.0, 2017.

Bastian Schürmann, Niklas Kochdumper, and Matthias Althoff. Reachset Model Predictive Control for Disturbed Nonlinear Systems. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 3463–3470, 2018.

Gautham Nayak Seetanadi, Karl-Erik Årzén, and Martina Maggio. Adaptive routing with guaranteed delay bounds using safe reinforcement learning. In *ACM Int. Conf. Proc. Series*, pp. 149 – 160, 2020.

Mahmoud Selim, Amr Alanwar, M Watheq El-Kharashi, Hazem M Abbas, and Karl H Johansson. Safe Reinforcement Learning using Data-Driven Predictive Control. In *Proc. of the Int. Conf. on Communications, Signal Processing, and their Applications (ICCSPA)*, pp. 1–6, 2022a.

Mahmoud Selim, Amr Alanwar, Shreyas Kousik, Grace Gao, Marco Pavone, and Karl H. Johansson. Safe Reinforcement Learning Using Black-Box Reachability Analysis. *IEEE Robotics and Automation Letters*, 7(4):10665–10672, 2022b.

Yifei Simon Shao, Chao Chen, Shreyas Kousik, and Ram Vasudevan. Reachability-Based Trajectory Safeguard (RTS): A Safe and Fast Reinforcement Learning Safety Layer for Continuous Control. *IEEE Robotics and Automation Letters*, 6(2):3663–3670, 2021.

Suhas Shyamsundar, Tommaso Mannucci, and Erik-Jan Van Kampen. Reinforcement learning based algorithm with Safety Handling and Risk Perception. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Joar Max Viktor Skalse, Nikolaus H R Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and Characterizing Reward Hacking. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, 2022.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID lagrangian methods. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 9133–9143, 2020.

Zachary Sunberg and Mykel Kochenderfer. Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces. *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 28(1): 259–263, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Leaning: An Introduction.* A Bradford Book, 2nd edition, 2018.

Daniel Tabas and Baosen Zhang. Computationally Efficient Safe Reinforcement Learning for Power Systems. In *Proc. of the American Control Conf. (ACC)*, pp. 3303–3310, 2022.

Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for Safety-Critical Control with Control Barrier Functions. In *Proc. of the Conf. on Learning for Dynamics and Control*, pp. 708–717, 2020.

Andrew J Taylor, Andrew Singletary, Yisong Yue, and Aaron D Ames. A Control Barrier Perspective on Episodic Learning via Projection-to-State Safety. *IEEE Control Systems Letters*, 5(3):1019–1024, 2021.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

Jakob Thumm and Matthias Althoff. Provably Safe Deep Reinforcement Learning for Robotic Manipulation in Human Environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 6344–6350, 2022.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite Markov decision processes with Gaussian processes. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*, pp. 4312–4320, 2016.

Stephen A Vavasis. *Complexity Theory: Quadratic Programming.* Springer, Boston, MA., 2001.

Kim Peter Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129(1):109597–109614, 2021.

Chengyu Wang, Luhan Wang, Zhaoming Lu, Xinghe Chu, Zhengrui Shi, Jiayin Deng, Tianyang Su, Guochu Shou, and Xiangming Wen. SRL-TR2: A Safe Reinforcement Learning Based TRajectory TRacker Framework. *IEEE Transactions on Intelligent Transportation Systems*, 24(6):5765–5780, 2023.

Linghao Wang, Miao Wang, and Yujun Zhang. A Safe Training Approach for Deep Reinforcement Learning-based Traffic Engineering. In *Proc. of the IEEE Int. Conf. on Communications (ICC)*, pp. 1450–1455. IEEE, 2022.

Xiao Wang. Ensuring Safety of Learning-Based Motion Planners Using Control Barrier Functions. *IEEE Robotics and Automation Letters*, 7(2):4773–4780, 2022.

Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proc. Volumes*, 40(12):462–467, 2007.

Chenchen Yang, Jing Liu, Haiying Sun, Junfeng Sun, Xiang Chen, and Lipeng Zhang. Safe Reinforcement Learning for CPSs via Formal Modeling and Verification. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, pp. 1–8, 2021.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, pp. 1–21, 2020.

Fei Ye, Shen Zhang, Pin Wang, and Ching Yao Chan. A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 1073–1080, 2021.

Peipei Yu, Hongcai Zhang, Yonghua Song, Hongxun Hui, and Ge Chen. District Cooling System Control for Providing Operating Reserve Based on Safe Deep Reinforcement Learning. *IEEE Transactions on Power Systems*, pp. 1–13, 2023.

Mario Zanon and Sebastien Gros. Safe Reinforcement Learning Using Robust MPC. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2021.

Melanie N Zeilinger, Davide M Raimondo, Alexander Domahidi, Manfred Morari, and Colin N Jones. On Real-Time Robust Model Predictive Control. *Automatica*, 50(3):683–694, 2014.

Jin Zhang, Yuxiang Guan, Liang Che, and Mohammad Shahidehpour. EV Charging Command Fast Allocation Approach based on Deep Reinforcement Learning with Safety Modules. *IEEE Transactions on Smart Grid*, 2023.

Wenshuai Zhao, Jorge Pena Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In *Proc. of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744, 2020.

A Appendix

MDP modification with action replacement

Action replacement alters the MDP on which the agent learns. Hunt et al. (2021) discuss this modification for discrete action spaces and uniformly sampling from the safe action space. We generalize this discussion to use any replacement function and also include continuous action spaces. We define $\boldsymbol{\psi}(\boldsymbol{s})$ so that it randomly samples the replacement action $\tilde{\boldsymbol{a}}$ according to a replacement policy $\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)$ with $\sum_{\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)=1$ for the discrete case, and $\int_{\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)d\tilde{\boldsymbol{a}}=1$ for the continuous case, and $\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)\geq 0\,\forall\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})$. In the example of uniform sampling from $\mathbb{A}_\varphi(\boldsymbol{s})$, the replacement policy is $\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)=1/|\mathbb{A}_\varphi(\boldsymbol{s})|$. By replacing unsafe actions, the transition function of the MDP changes to

$$T_\varphi(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')=\begin{cases}T(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}'), & \text{if } \varphi(\boldsymbol{s},\boldsymbol{a})=1\\ T_r(\boldsymbol{s},\boldsymbol{s}'), & \text{otherwise,}\end{cases} \tag{15}$$

$$T_r(\boldsymbol{s},\boldsymbol{s}')=\sum_{\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)T(\boldsymbol{s},\tilde{\boldsymbol{a}},\boldsymbol{s}'). \tag{16}$$

The reward function of the MDP changes accordingly to

$$r_\varphi(\boldsymbol{s},\boldsymbol{a})=\begin{cases}r(\boldsymbol{s},\boldsymbol{a}), & \text{if } \varphi(\boldsymbol{s},\boldsymbol{a})=1\\ r_r(\boldsymbol{s}), & \text{otherwise,}\end{cases} \tag{17}$$

$$r_r(\boldsymbol{s})=\sum_{\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)r(\boldsymbol{s},\tilde{\boldsymbol{a}}). \tag{18}$$

In the continuous case, we get $T_r(\boldsymbol{s},\boldsymbol{s}')$ by marginalizing the transition probability density function over $\mathbb{A}_\varphi(\boldsymbol{s})$

$$T_r(\boldsymbol{s},\boldsymbol{s}')=\int_{\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)T(\boldsymbol{s},\tilde{\boldsymbol{a}},\boldsymbol{s}')d\tilde{\boldsymbol{a}}, \tag{19}$$

and $r_r(\boldsymbol{s})$ analogously

$$r_r(\boldsymbol{s})=\int_{\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r\left(\tilde{\boldsymbol{a}}|\boldsymbol{s}\right)r(\boldsymbol{s},\tilde{\boldsymbol{a}})d\tilde{\boldsymbol{a}}. \tag{20}$$

Environment Parameters

We give an overview of all environment-specific parameters in Table 3 and Table 4.

Table 3: Pendulum environment parameters.

Parameter	Value
Gravity g	$9.81\,\mathrm{m\,s^{-2}}$
Mass m	$1\,\mathrm{kg}$
Length l	$1\,\mathrm{m}$

Table 4: 2D quadrotor environment parameters.

Parameter	Value
Gravity g	$9.81\,\mathrm{m\,s^{-2}}$
k	$1\,1/\mathrm{kg}$
d_0	70
$d1$	17
n_0	55
$\mathbb{W}$	$[[-0.1, 0.1], [-0.1, 0.1]]$

Hyperparameters for learning algorithms

We specify the hyperparameters for all learning algorithms (see Table 5 for PPO, Table 6 for TD3, and Table 7 for DQN) that are different from the Stable Baselines3 (Raffin et al., 2021) default values. Additionally, the code for the experiments is submitted as supplementary material to further increase the transparency.

Table 5: Hyperparameters for PPO.

Parameter	Pendulum	2D quadrotor
Learning rate	$1\mathrm{E}-4$	$5\mathrm{E}-5$
Discount factor γ	0.98	0.999
Steps per update	2048	512
Optimization epochs	20	30
Minibatch size	16	128
Max gradient clipping	0.9	0.5
Entropy coefficient	$1\mathrm{E}-3$	$2\mathrm{E}-6$
Value function coefficient	0.045	0.5
Clipping range	0.3	0.1
GAE λ	0.8	0.92
Activation function	ReLU	ReLU
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	100k

Full evaluation

In this section, we present all training results of the five RL algorithms TD3, SAC, DQN, and PPO continuous and discrete. We compare these algorithms on the inverted pendulum and 2D quadrotor environment on ten random seeds. The tested algorithms are action replacement with $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ and $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$, action projection with the CBFs implementation, and action masking. When action replacement is used with a failsafe controller, we omit *safe action* and *both* because, especially in the discrete action space, the failsafe controller might use an action that is not in the action space.

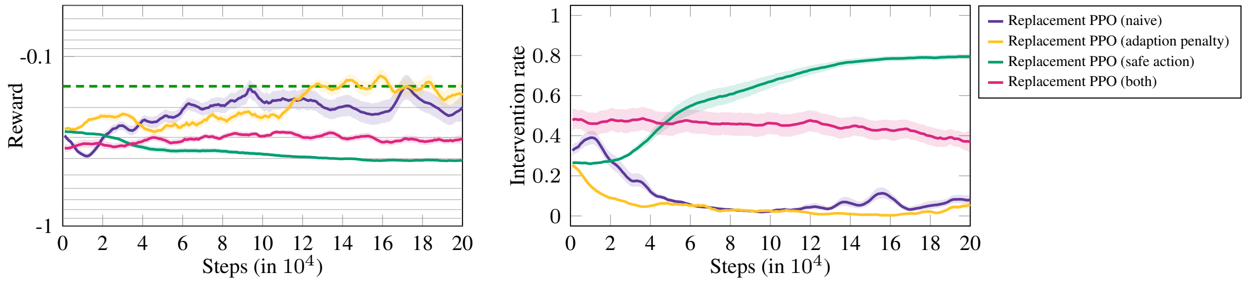
Table 6: Hyperparameters for TD3.

Parameter	Pendulum	2D quadrotor
Learning rate	3.5E − 3	2E − 3
Replay buffer size	1E4	1E5
Discount factor γ	0.98	0.98
Initial exploration steps	10E3	100
Steps between model updates	256	5
Gradient steps per model update	256	10
Minibatch size per gradient step	512	512
Soft update coefficient τ	5E − 3	5E − 3
Gaussian smoothing noise σ	0.2	0.12
Activation function	ReLU	ReLU
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	100k

Table 7: Hyperparameters for DQN.

Parameter	Pendulum	2D quadrotor
Learning rate	2E − 3	1E − 4
Replay buffer size	5E4	1E6
Discount factor γ	0.95	0.999 99
Initial exploration steps	500	100
Steps between model updates	8	2
Gradient steps per model update	4	4
Minibatch size per gradient step	512	64
Maximum for gradient clipping	10	100
Update frequency target network	1E3	1E3
Initial exploration probability ϵ	1.0	0.137
Linear interpolation steps of ϵ	6E3	1E4
Final exploration probability ϵ	0.1	0.004
Activation function	Tanh	Tanh
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	100k

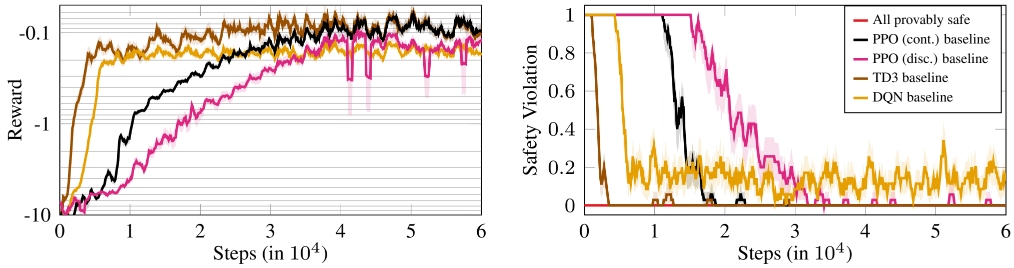
First, we present the effect of the learning tuples on the on-policy algorithm PPO in figure 4. This comparison clearly shows the negative effect of the *safe action* tuple on the training performance of PPO.



(a) Reward action replacement – PPO

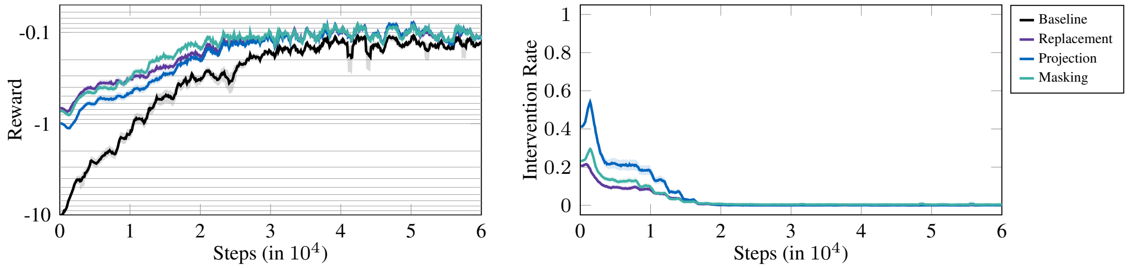
(b) Intervention rate action replacement – PPO

Figure 4: Evaluation of the training tuples for the 2D quadrotor averaged over the continuous and discrete PPO implementation using action replacement with ten random seeds each. The left column depicts the reward and the right column the safety intervention rate.



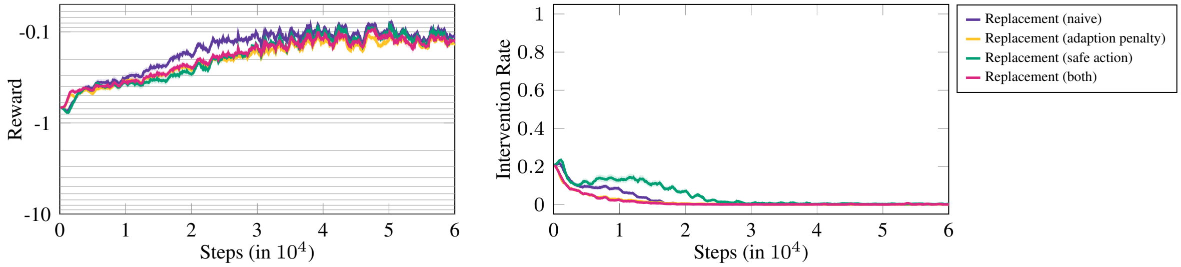
(a) Reward baselines

(b) Safety violation rate baselines



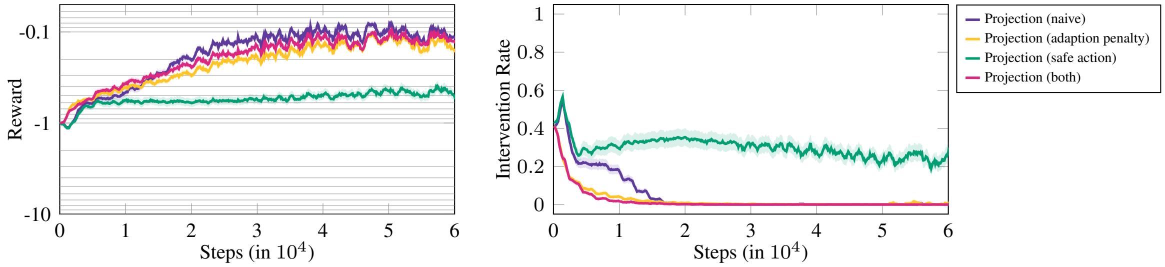
(c) Reward provably safe RL algorithms

(d) Intervention rate provably safe RL algorithms



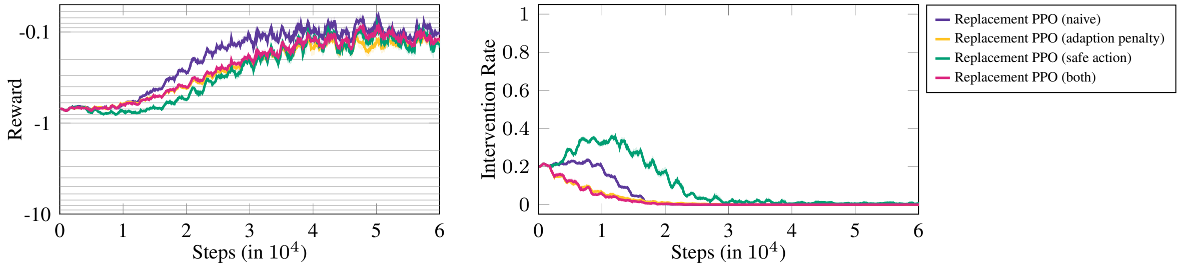
(e) Reward action replacement

(f) Intervention rate action replacement



(g) Reward action projection

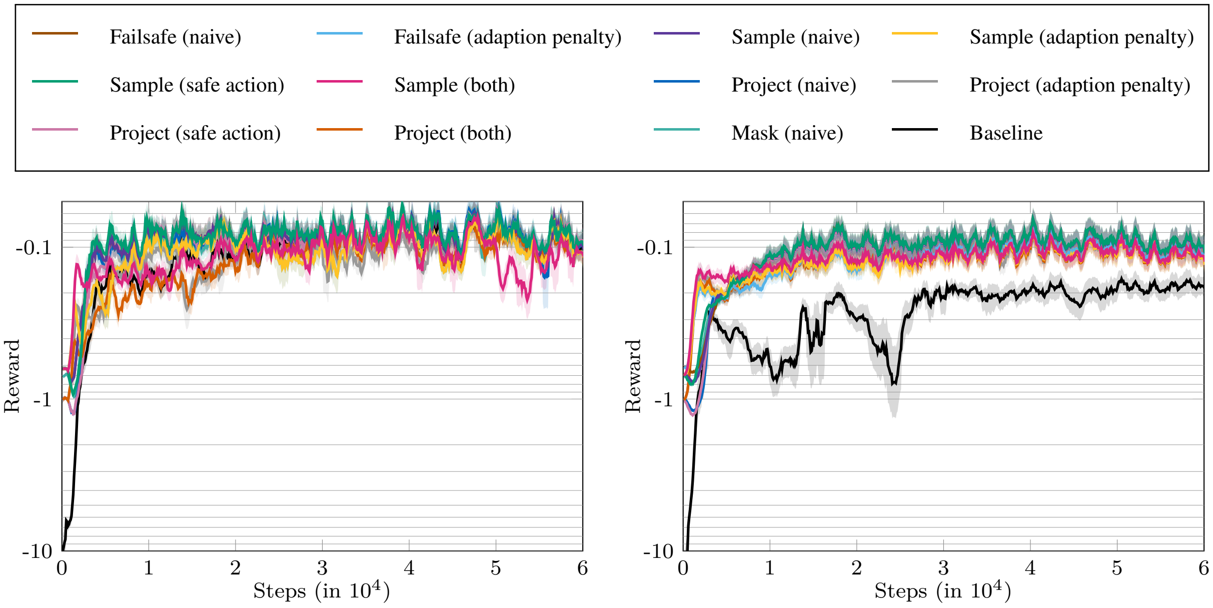
(h) Intervention rate action projection



(i) Reward action replacement – PPO

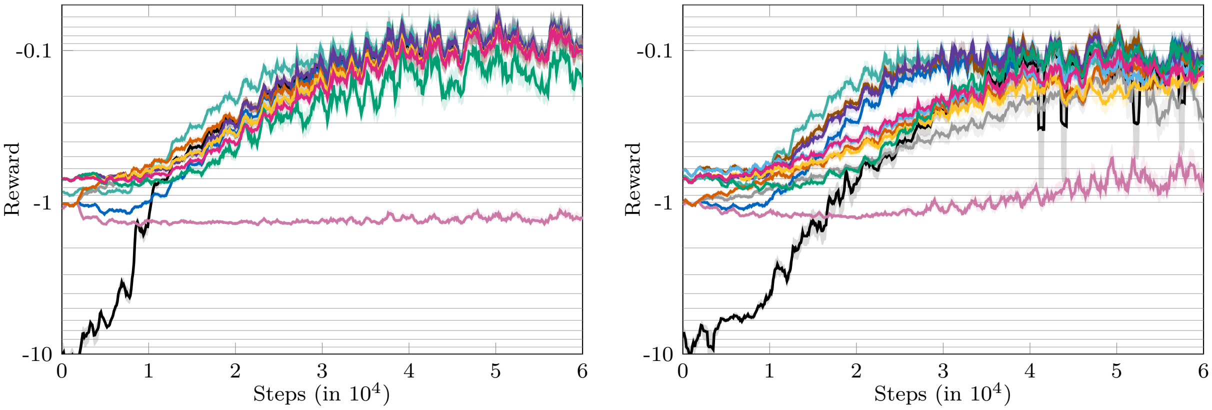
(j) Intervention rate action replacement – PPO

Figure 5: Evaluation of the training tuples for the pendulum averaged over ten random seeds each. The left column depicts the reward and the right column the safety intervention rate. Please refer to figures 2 and 3 for the regarding 2D quadrotor results.



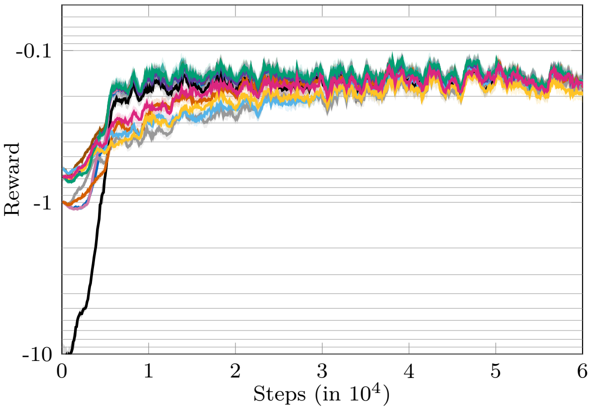
(a) TD3

(b) SAC



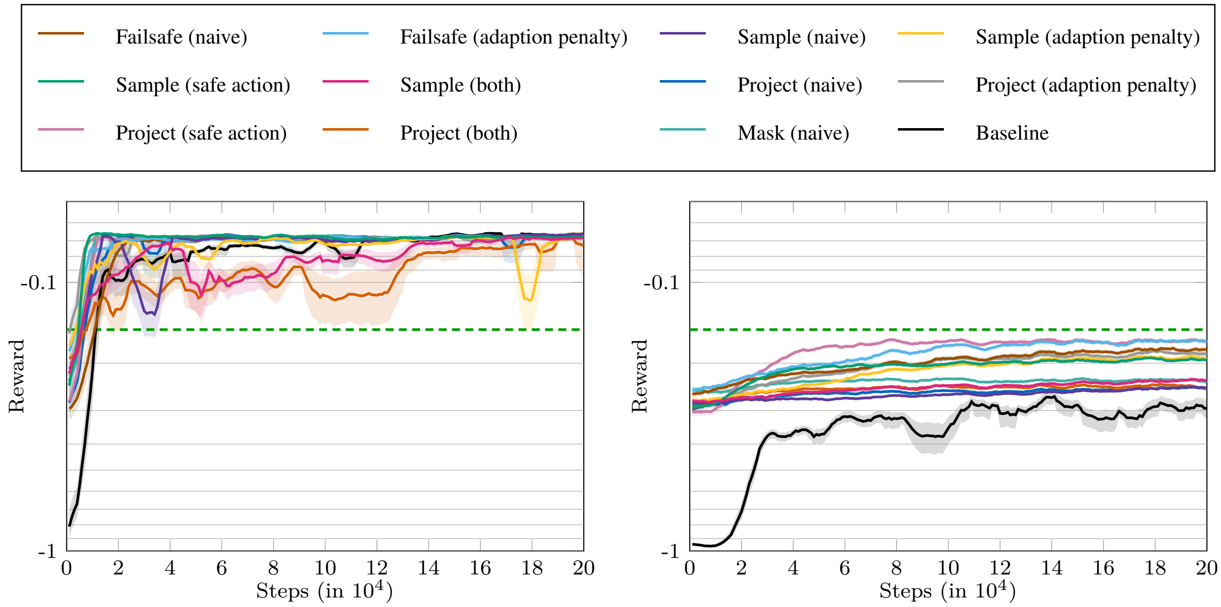
(c) PPO continuous

(d) PPO discrete



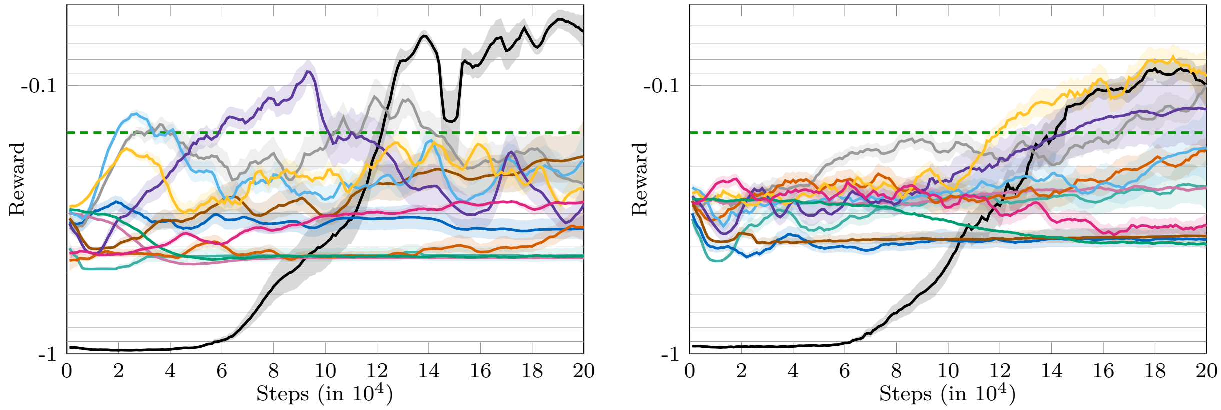
(e) DQN

Figure 6: **Pendulum:** Average reward and standard deviation per training step for TD3, SAC, DQN, PPO discrete, and PPO continuous. For each configuration, five training runs with different random seeds were conducted. Each subplot contains all implemented variants. Note that for better comparability the reward for the *adaption penalty* variants is still r and the adaption penalty r^* is not included in the curves.



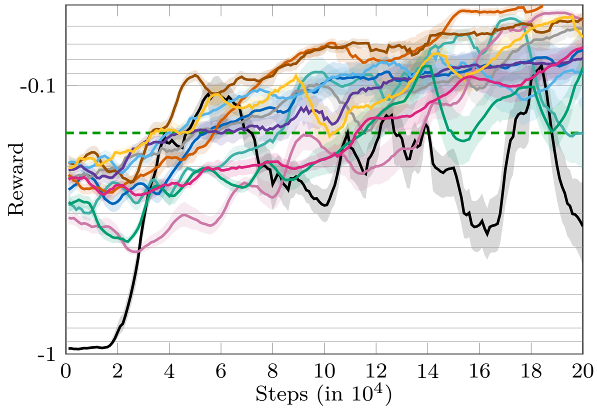
(a) TD3

(b) SAC



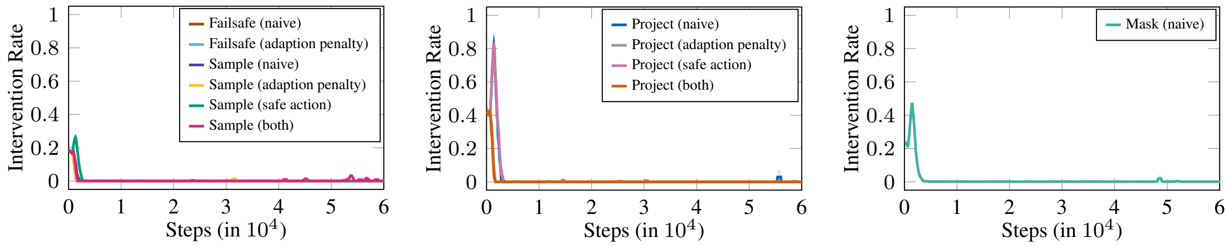
(c) PPO continuous

(d) PPO discrete

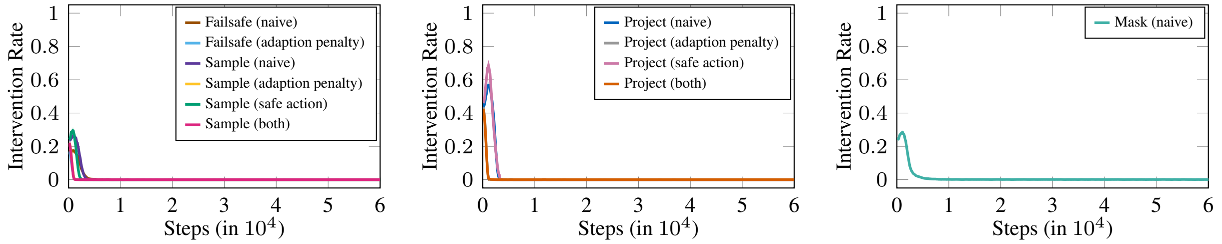


(e) DQN

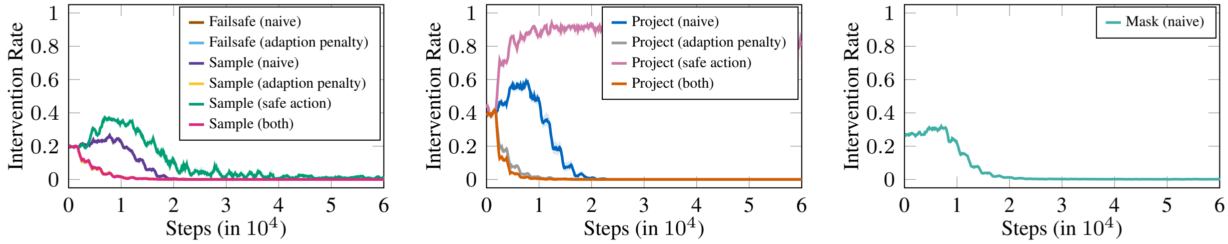
Figure 7: **2D quadrotor:** Average reward and standard deviation per training step for TD3, SAC, DQN, PPO discrete, and PPO continuous. For each configuration, five training runs with different random seeds were conducted. Each subplot contains all implemented variants. Note that for better comparability the reward for the *adaption penalty* variants is still r and the adaption penalty r^* is not included in the curves.



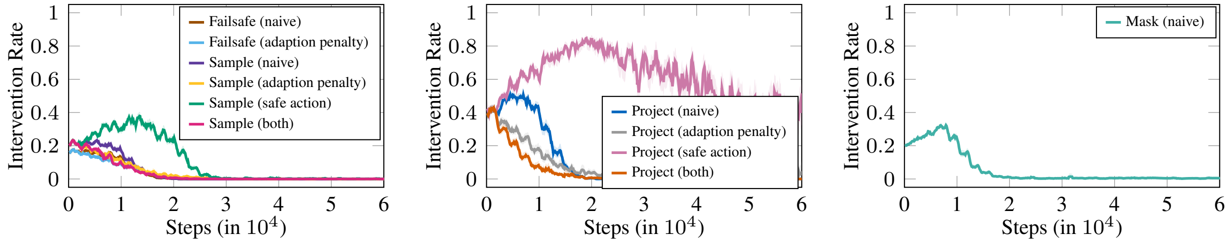
(a) Replacement TD3 (b) Projection TD3 (c) Masking TD3



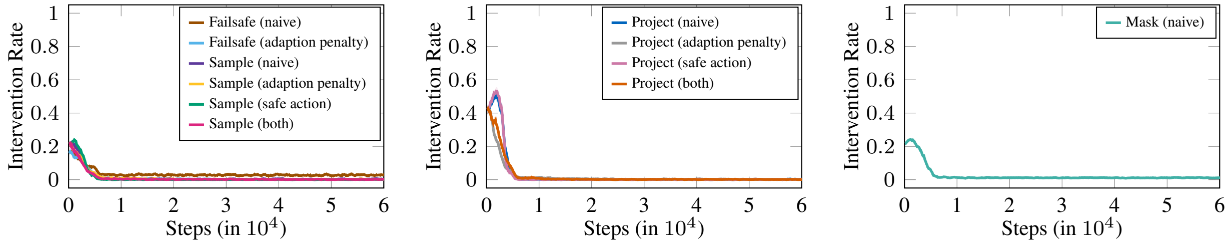
(d) Replacement SAC (e) Projection SAC (f) Masking SAC



(g) Replacement PPO cont. (h) Projection PPO cont. (i) Masking PPO cont.

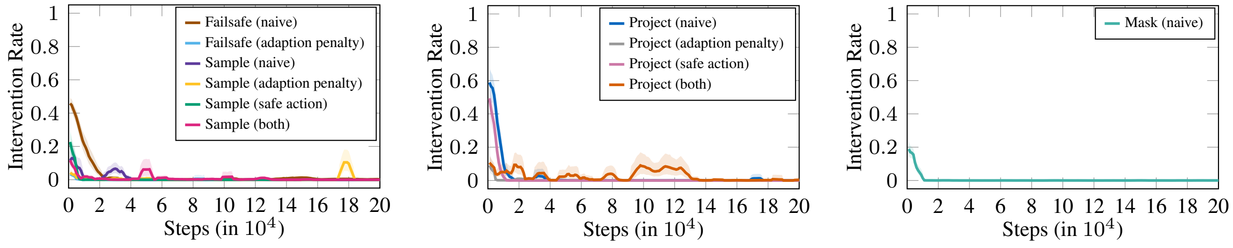


(j) Replacement PPO disc. (k) Projection PPO disc. (l) Masking PPO disc.

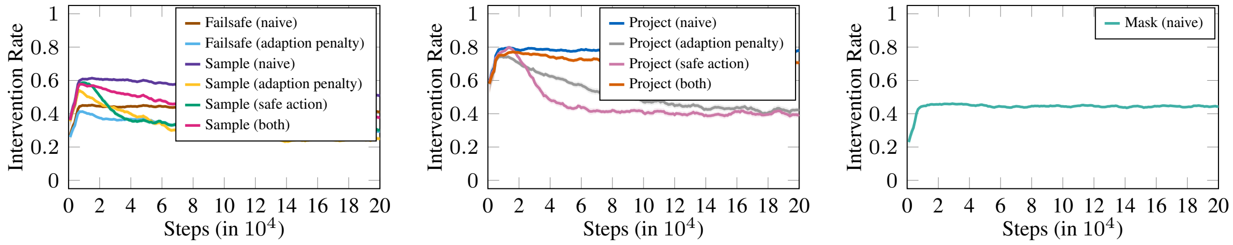


(m) Replacement DQN (n) Projection DQN (o) Masking DQN

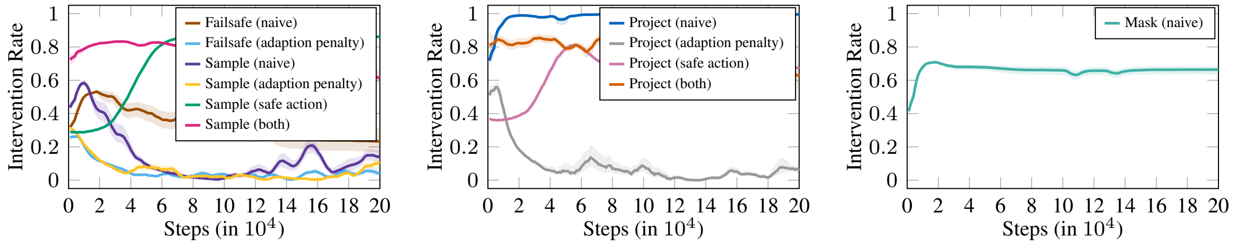
Figure 8: **Pendulum:** Intervention rate for TD3, SAC, DQN, PPO discrete, and PPO continuous. Fig. (a) - (c) show the intervention rate of TD3 agents, (d) - (f) for DQN agents, (g) - (i) for discrete PPO agents, and (j) - (l) for continuous PPO agents.



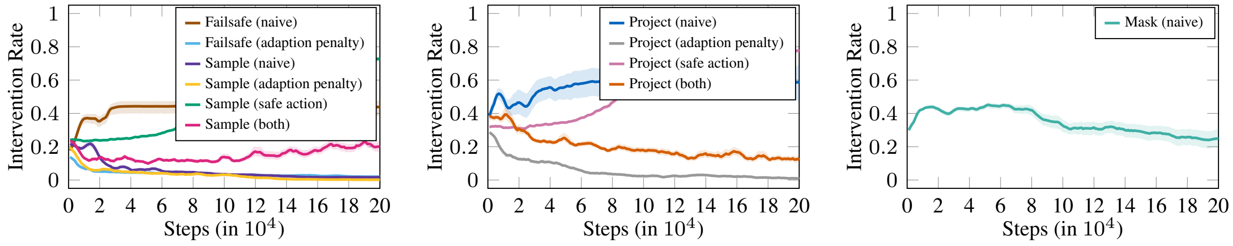
(a) Replacement TD3 (b) Projection TD3 (c) Masking TD3



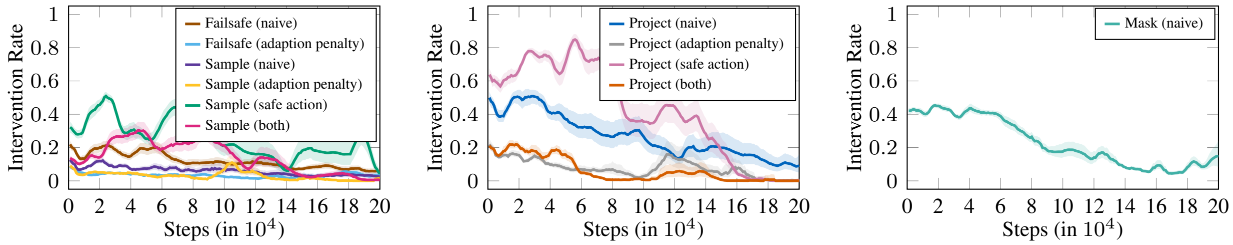
(d) Replacement SAC (e) Projection SAC (f) Masking SAC



(g) Replacement PPO cont. (h) Projection PPO cont. (i) Masking PPO cont.



(j) Replacement PPO disc. (k) Projection PPO disc. (l) Masking PPO disc.



(m) Replacement DQN (n) Projection DQN (o) Masking DQN

Figure 9: **2D quadrotor:** Intervention rate for TD3, SAC, DQN, PPO discrete, and PPO continuous. Fig. (a) - (c) show the intervention rate of TD3 agents, (d) - (f) for DQN agents, (g) - (i) for discrete PPO agents, and (j) - (l) for continuous PPO agents.

Table 8: Mean and standard deviation of 15 pendulum deployment episodes.

Approach	**Reward**		**Intervention Rate**		**Safety Violation**	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
PPO (continuous)						
Projection (SafeAction)	-1.14	0.42	0.76	0.29	0.000	0.000
Projection (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Projection (Both)	-0.07	0.07	0.00	0.00	0.000	0.000
Projection (Naive)	-0.06	0.07	0.00	0.00	0.000	0.000
Sample (SafeAction)	-0.13	0.16	0.01	0.02	0.000	0.000
Sample (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Sample (Both)	-0.07	0.07	0.00	0.00	0.000	0.000
Sample (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
FailSafe (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
FailSafe (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
Baseline (Naive)	-0.06	0.07	—	—	0.000	0.000
Masking (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Masking (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
PPO (discrete)						
Projection (SafeAction)	-0.52	0.55	0.28	0.42	0.000	0.000
Projection (AdaptionPenalty)	-0.14	0.12	0.00	0.00	0.000	0.000
Projection (Both)	-0.09	0.09	0.00	0.00	0.000	0.000
Projection (Naive)	-0.15	0.27	0.03	0.10	0.000	0.000
Sample (SafeAction)	-0.09	0.08	0.00	0.00	0.000	0.000
Sample (AdaptionPenalty)	-0.13	0.16	0.00	0.00	0.000	0.000
Sample (Both)	-0.10	0.08	0.00	0.00	0.000	0.000
Sample (Naive)	-0.09	0.08	0.00	0.00	0.000	0.000
FailSafe (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.000	0.000
FailSafe (Naive)	-0.08	0.07	0.00	0.00	0.000	0.000
Baseline (Naive)	-0.08	0.07	—	—	0.000	0.000
Masking (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Masking (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
TD3						
Projection (SafeAction)	-0.07	0.07	0.00	0.00	0.000	0.000
Projection (AdaptionPenalty)	-0.08	0.08	0.00	0.00	0.000	0.000
Projection (Both)	-0.07	0.07	0.00	0.00	0.000	0.000
Projection (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
Sample (SafeAction)	-0.07	0.07	0.00	0.00	0.000	0.000
Sample (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.000	0.000
Sample (Both)	-0.09	0.07	0.00	0.00	0.000	0.000
Sample (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
FailSafe (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.000	0.000
FailSafe (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
Baseline (Naive)	-0.07	0.07	—	—	0.000	0.000
Masking (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Masking (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
DQN						
Projection (SafeAction)	-0.07	0.07	0.00	0.00	0.000	0.000
Projection (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Projection (Both)	-0.07	0.08	0.00	0.00	0.000	0.000
Projection (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
Sample (SafeAction)	-0.07	0.07	0.00	0.00	0.000	0.000
Sample (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.000	0.000
Sample (Both)	-0.07	0.08	0.00	0.00	0.000	0.000
Sample (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
FailSafe (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
FailSafe (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000
Baseline (Naive)	-0.07	0.07	—	—	0.000	0.000
Masking (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.000	0.000
Masking (Naive)	-0.07	0.07	0.00	0.00	0.000	0.000

Table 9: Mean and standard deviation of 15 2D Quadrotor deployment episodes.

Approach	**Reward**		**Intervention Rate**		**Safety Violation**	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
PPO (continuous)						
Projection (SafeAction)	-0.44	0.00	0.68	0.00	0.000	0.000
Projection (AdaptionPenalty)	-0.33	0.07	0.44	0.05	0.000	0.000
Projection (Both)	-0.39	0.07	0.57	0.13	0.000	0.000
Projection (Naive)	-0.31	0.10	0.47	0.25	0.000	0.000
Sample (SafeAction)	-0.43	0.00	0.86	0.00	0.000	0.000
Sample (AdaptionPenalty)	-0.28	0.12	0.10	0.12	0.000	0.000
Sample (Both)	-0.36	0.03	0.41	0.20	0.000	0.000
Sample (Naive)	-0.39	0.06	0.23	0.13	0.000	0.000
FailSafe (AdaptionPenalty)	-0.31	0.11	0.08	0.06	0.000	0.000
FailSafe (Naive)	-0.27	0.11	0.27	0.29	0.000	0.000
Baseline (Naive)	-0.86	0.01	—	—	0.941	0.009
Masking (AdaptionPenalty)	-0.43	0.09	0.57	0.28	0.000	0.000
Masking (Naive)	-0.43	0.09	0.57	0.28	0.000	0.000
PPO (discrete)						
Projection (SafeAction)	-0.44	0.01	0.74	0.21	0.000	0.000
Projection (AdaptionPenalty)	-0.24	0.13	0.02	0.02	0.000	0.000
Projection (Both)	-0.35	0.11	0.16	0.14	0.000	0.000
Projection (Naive)	-0.34	0.14	0.46	0.37	0.000	0.000
Sample (SafeAction)	-0.42	0.02	0.82	0.01	0.000	0.000
Sample (AdaptionPenalty)	-0.11	0.10	0.00	0.00	0.000	0.000
Sample (Both)	-0.38	0.09	0.32	0.18	0.000	0.000
Sample (Naive)	-0.13	0.16	0.02	0.03	0.000	0.000
FailSafe (AdaptionPenalty)	-0.19	0.15	0.01	0.03	0.000	0.000
FailSafe (Naive)	-0.34	0.13	0.41	0.21	0.000	0.000
Baseline (Naive)	-0.11	0.03	—	—	0.000	0.000
Masking (AdaptionPenalty)	-0.25	0.14	0.28	0.21	0.000	0.000
Masking (Naive)	-0.25	0.14	0.28	0.21	0.000	0.000
TD3						
Projection (SafeAction)	-0.21	0.03	0.28	0.10	0.000	0.000
Projection (AdaptionPenalty)	-0.22	0.03	0.26	0.10	0.000	0.000
Projection (Both)	-0.21	0.04	0.28	0.12	0.000	0.000
Projection (Naive)	-0.25	0.05	0.26	0.17	0.000	0.000
Sample (SafeAction)	-0.18	0.03	0.07	0.01	0.000	0.000
Sample (AdaptionPenalty)	-0.21	0.04	0.13	0.05	0.000	0.000
Sample (Both)	-0.21	0.06	0.06	0.03	0.000	0.000
Sample (Naive)	-0.19	0.04	0.12	0.09	0.000	0.000
FailSafe (AdaptionPenalty)	-0.20	0.03	0.05	0.02	0.000	0.000
FailSafe (Naive)	-0.26	0.09	0.05	0.02	0.000	0.000
Baseline (Naive)	-0.90	0.05	—	—	0.945	0.018
Masking (AdaptionPenalty)	-0.16	0.03	0.03	0.03	0.000	0.000
Masking (Naive)	-0.16	0.03	0.03	0.03	0.000	0.000
DQN						
Projection (SafeAction)	-0.06	0.02	0.00	0.01	0.000	0.000
Projection (AdaptionPenalty)	-0.05	0.00	0.00	0.00	0.000	0.000
Projection (Both)	-0.05	0.01	0.00	0.00	0.000	0.000
Projection (Naive)	-0.09	0.06	0.12	0.24	0.000	0.000
Sample (SafeAction)	-0.07	0.01	0.01	0.02	0.000	0.000
Sample (AdaptionPenalty)	-0.06	0.03	0.00	0.00	0.000	0.000
Sample (Both)	-0.07	0.02	0.00	0.00	0.000	0.000
Sample (Naive)	-0.05	0.01	0.00	0.00	0.000	0.000
FailSafe (AdaptionPenalty)	-0.06	0.02	0.02	0.04	0.000	0.000
FailSafe (Naive)	-0.07	0.03	0.10	0.10	0.000	0.000
Baseline (Naive)	-0.24	0.37	—	—	0.199	0.398
Masking (AdaptionPenalty)	-0.15	0.16	0.14	0.26	0.000	0.000
Masking (Naive)	-0.15	0.16	0.14	0.26	0.000	0.000