BEYOND THE CONSTRAINT: PROBING CONSTRAINED RL BOTTLENECKS WITH PHASIC POLICY GRADIENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Constrained Reinforcement Learning (CRL) extends policy optimization to Constrained Markov Decision Processes (CMDPs), where agents must maximize reward while satisfying explicit cost budgets. Although existing on-policy CRL methods differ in how they enforce constraints, they typically follow the same training paradigm where collected experience is used once and then discarded. We investigate whether this practice underutilizes information contained in sampled CRL trajectories, which provide supervision through both reward and cost signals. To this end, we introduce a phasic training framework for CRL that separates policy optimization from a dedicated auxiliary phase. During this phase, retained rollouts are reused to refine reward and cost representations that are then distilled into the policy network. The framework acts as a modular wrapper and can extend existing CRL algorithms without modifying their constraint-handling mechanisms. Experiments on Safety Gymnasium show that our extension to CPO, CPPO-PID and TRPO-PID yields consistent improvements in sample efficiency and cumulative reward in environments with rich observation structure. This suggests that representation learning and data utilization are important and underexplored factors in on-policy CRL performance.

1 INTRODUCTION

Real-world decision-making problems are rarely defined solely by what an agent should achieve, but also by what it must avoid. In many applications, undesirable events such as collisions, excessive forces, unsafe velocities, or resource violations are not merely negative outcomes to be traded against reward, but operational requirements that must remain within prescribed limits. Constrained Reinforcement Learning (CRL) integrates these hard limits into the learning process of Reinforcement Learning (RL) by decoupling task performance from safety requirements. By formalizing constraints as explicit cost budgets, CRL allows agents to maximize cumulative reward while keeping expected cumulative cost below a specified threshold. This makes constraint satisfaction an explicit part of both the optimization problem and the evaluation criterion, which is essential in domains such as robotics and industrial control (García & Fernández, 2015; Gu et al., 2022).

On-policy CRL research has predominantly focused on how constraints should be enforced during policy optimization. Constrained Policy Optimization (CPO) (Achiam et al., 2017) bounds the potential increase in cost within a second-order approximated KL trust region. TRPO-PID (Stooke et al., 2020) retains TRPO's (Schulman et al., 2015) second-order trust region but handles constraints through a PID-controlled Lagrangian multiplier, while CPPO-PID (Stooke et al., 2020) replaces the second-order trust-region update with PPO's clipped first-order surrogate (Schulman et al., 2017). Together, these methods span error-bound-based and penalty-based constraint handling as well as second- and first-order policy updates.

Despite these architectural differences, all three share the same basic training paradigm: experience is collected under the current policy, used to update actor and critics, and then discarded. This single-pass usage of data may not fully exploit the information contained in such trajectories. In addition to reward supervision, CRL provides a cost signal that may describe complementary aspects of the environment, such as unsafe regions, constraint boundaries, or undesirable transitions. The reward and cost critics may hence encode different but jointly useful properties.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This observation connects CRL to a broader question in deep reinforcement learning: *To what extent is performance determined by the optimization algorithm itself, and to what extent by the quality of the learned representation?* Standard end-to-end reinforcement-learning updates can underutilize the representational content of collected experience (Stooke et al., 2021). In prior work, Phasic Policy Gradient (PPG) addressed this problem by separating on-policy policy optimization from a subsequent auxiliary phase in which retained trajectories are reused for representation learning (Cobbe et al., 2021). During this auxiliary phase, value-related information is distilled into the policy network while a KL-penalty limits changes to the policy. The additional cost channel makes this question especially interesting for CRL: Constrained trajectories provide supervision not only through reward returns, but also through costs that are often only weakly aligned with reward, but still strongly relate to environment dynamics.

We hence propose a phasic training scheme for on-policy CRL. As in PPG we alternate between a policy and an auxiliary phase. During the policy phase, the underlying CRL algorithm performs its actor and critic updates without modification, while states and their reward and cost value targets are retained. During the auxiliary phase, stored targets are used to train reward and cost auxiliary heads attached to the policy network, thereby providing additional reward and cost supervision. At the same time, the reward and cost critics are further trained on the buffered data.

Our framework wraps around existing CRL algorithms and leaves constraint-handling of the backbone method unchanged. We showcase three representative algorithms: CPO, TRPO-PID, and CPPO-PID. As the scale of the auxiliary value losses can vary substantially across environments and over the course of training, we replace the fixed KL cloning coefficient used in PPG with an adaptive Lagrange multiplier that enforces a target policy-space divergence during the auxiliary phase. Our contributions are as follows:

1. We introduce a modular phasic framework for on-policy constrained reinforcement learning that reuses collected trajectories for reward- and cost-aware auxiliary learning without modifying the backbone algorithm's constraint-handling mechanism,
2. We formulate policy preservation during the auxiliary phase as an adaptive KL constraint, avoiding the environment-dependent behavior of a fixed cloning-loss coefficient, and
3. Across CPO, CPPO-PID, and TRPO-PID, we show consistent reward improvements in object-centric Safety Gymnasium navigation tasks, while the negative results on `HalfCheetahVelocity` characterize an important limitation of the approach.

Our results suggest that data utilization and auxiliary representation learning are important yet underexplored factors in on-policy CRL performance. They also indicate that the value of phasic training depends on the structure of the observations and on when reward and cost signals become informative during learning, as well as the distribution of cost and reward across states.

2 Background

2.1 Constrained Markov Decision Processes

Constrained RL augments the standard Markov Decision Process with explicit cost signals and corresponding budgets. We consider a Constrained Markov Decision Process (CMDP) (Altman, 1999), in which a policy π maximizes its expected discounted return while satisfying an expected discounted cost constraint:

$$\max_{\pi} \ \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \quad \text{s.t.} \quad \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t c_t\right] \le d, \tag{1}$$

where r_t and c_t denote the reward and cost signals at time step t, γ is the discount factor, and d is the prescribed cost budget.

For continuous control problems, on-policy CRL methods commonly follow an actor–critic formulation. A parameterized policy π_θ selects actions, while a reward critic $V^R_{\theta_R}$ estimates expected cumulative reward and provides a variance-reducing baseline for the policy update. Constrained methods additionally maintain a cost critic $V^C_{\theta_C}$, which estimates expected cumulative cost and supplies the information required to enforce or regulate the constraint. Given a rollout, we can estimate regression targets $\hat{V}_t^{R,\text{targ}}$ and $\hat{V}_t^{C,\text{targ}}$ for values and costs respectively.

Reward and cost critics may capture complementary aspects of the environment. The reward critic represents task-relevant structure, whereas the cost critic may emphasize unsafe regions, constraint boundaries, or transitions that contribute to future violations. However, in standard on-policy CRL, both critics and the policy are trained on recent experience, after which the trajectories are discarded.

Different CRL algorithms use reward- and cost-related estimates in different ways. Error-bound-based methods directly restrict the policy update using a local approximation of the cost constraint (Achiam et al., 2017), whereas Lagrangian methods incorporate the estimated cost into a penalized policy objective (Stooke et al., 2020; Schulman et al., 2015). The framework proposed in this work does not alter these algorithm-specific actor updates. Instead, it wraps the policy update of the underlying CRL algorithm inside a two-phase approach for reusing previously collected data.

2.2 Phasic Policy Optimization

Standard on-policy actor–critic methods impose the same data-use schedule on components with different learning requirements. Policy optimization requires sufficiently recent on-policy samples to obtain reliable policy gradients (Cobbe et al., 2021). Critics and learned representations, in contrast, can often benefit from additional optimization on a broader diversity of previously visited states (Wang et al., 2023). Discarding trajectories after a single sequence of updates may therefore leave useful representational information unexploited.

Phasic Policy Gradient (PPG) (Cobbe et al., 2021) addresses this mismatch by dividing training into two alternating stages: a policy phase and an auxiliary phase. During the policy phase, the policy and value function are optimized using freshly sampled on-policy trajectories. Rather than immediately discarding these trajectories, PPG retains their observations and value targets in a buffer over multiple policy iterations. During the subsequent auxiliary phase, the buffered data is reused to train an auxiliary value head attached to the policy network. Although the output of this head is not required for action selection, its gradients update the representation shared with the policy. This allows the policy network to benefit from value-related supervision without requiring the policy-gradient and value-regression objectives to be optimized jointly during the policy phase.

Because the auxiliary objective modifies parameters that also determine the action distribution, unconstrained auxiliary optimization could substantially change the policy. PPG therefore preserves the policy obtained at the end of the policy phase through a cloning objective. Let π_{old} be the policy before the auxiliary update. The auxiliary objective can be written as

$$L^{\mathrm{PPG}} = L^{\mathrm{aux_reward}} + \beta_{\mathrm{clone}} \cdot \hat{\mathbb{E}}_t \left[D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(\cdot|s_t) \,\|\, \pi_{\theta_\pi}(\cdot|s_t)) \right], \tag{2}$$

where $L^{\mathrm{aux_reward}}$ trains the policy network's auxiliary value head and β_{clone} controls the trade-off between auxiliary learning and policy preservation.

PPG was originally developed for unconstrained reinforcement learning and uses a single value-related auxiliary signal. In a CMDP, each trajectory provides two forms of supervision: reward returns and cost returns. These signals may differ substantially in scale, sparsity, temporal structure, and relevance across the state space. Extending phasic training to constrained reinforcement learning therefore requires determining how reward- and cost-related information should supervise the policy representation and how auxiliary updates can be performed without destabilizing the policy produced by the underlying constrained optimization algorithm.

3 Method

We sketch our universal augmentation scheme in Figure 1. Our method uses an arbitrary CRL policy (such as CPPO-PID, TRPO-PID or CPO) together with a value distillation phase. First, we perform the standard (on-policy) update as is described in the respective CRL method. After the policy is trained, but before the subsequent auxiliary phase begins, the current policy is frozen as $\pi_{\theta_{\mathrm{old}}}$, establishing a fixed reference point for the KL penalty that bounds policy drift during distillation. Second, the auxiliary phase then runs for E_{aux} epochs over the full buffer, optimizing a joint objective that couples representation distillation with policy stability:

$$L^{\mathrm{joint}} = L^{\mathrm{aux_reward}} + \alpha_{\mathrm{cost}} \cdot L^{\mathrm{aux_cost}} + \beta_{\mathrm{clone}} \cdot \hat{\mathbb{E}}_t \left[D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(\cdot|s_t) \,\|\, \pi_{\theta_\pi}(\cdot|s_t)) \right], \tag{3}$$

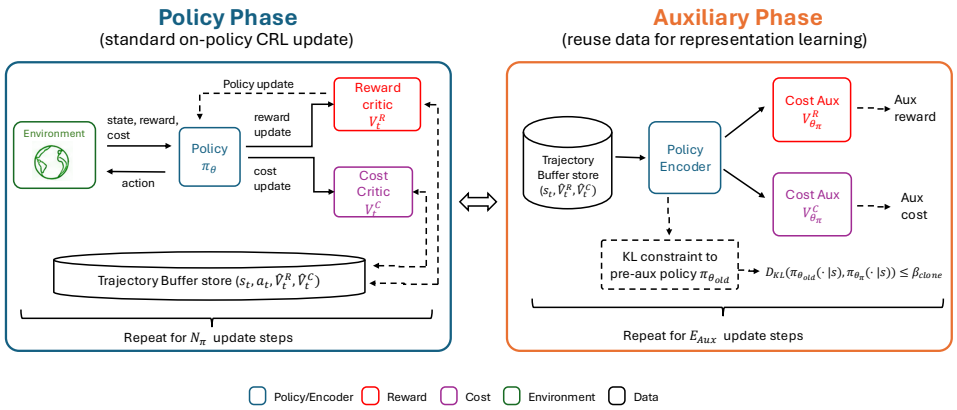


Figure 1: Our method is split into two components: First a policy phase (left) which can correspond to any actor-critic method. This component needs no changes and directly corresponds to the original implementation. The second phase (right) is our independent auxiliary phase which keeps the policy stationary by enforcing minimal change in the policy head, while improving the policy encoder by distilling value and cost information into the policy backbone.

where $L^{\text{aux_reward}}$ and $L^{\text{aux_cost}}$ are MSE losses computed from the output of the policy network's auxiliary heads:

$$L^{\text{aux_reward}} = \hat{\mathbb{E}}_t\left[\left(V_{\theta_\pi}^R(s_t) - \hat{V}_t^{R,\text{targ}}\right)^2\right], \text{ and } L^{\text{aux_cost}} = \hat{\mathbb{E}}_t\left[\left(V_{\theta_\pi}^C(s_t) - \hat{V}_t^{C,\text{targ}}\right)^2\right]. \quad (4)$$

These auxiliary value heads of the policy network as displayed in Figure 1 (right) define the interface used to transfer reward- and cost-aware information learned by the separate critics into the actor. While being in the policy phase, the underlying on-policy algorithms only use the network parameterized by θ_π through its action distribution $\pi_{\theta_\pi}(\cdot \mid s)$, during the auxiliary phase the additional heads, $V_{\theta_\pi}^R(s)$ and $V_{\theta_\pi}^C(s)$ that correspond to reward and cost value predictions become active.

The hyperparameter α_{cost} is motivated by the observation that scales of cost- and reward-targets tend to differ significantly. An unweighted auxiliary loss would therefore not necessarily reflect the intended balance between reward- and cost-related representation learning. Beyond this scaling role, α_{cost} also lets us control the relative influence of cost-value information during distillation, allowing us to study which balance between reward- and cost-related representations is most beneficial for the underlying learner (see Figures 6 to 8 in Section D and Section C).

In contrast to the original PPG formulation, which uses a fixed β_{clone} coefficient to penalize the divergence of π_{θ_π} from $\pi_{\theta_{\text{old}}}$ during the auxiliary update, we utilize a much more flexible adaptive Lagrange multiplier that enforces a target KL budget during the auxiliary phase. Concretely, we constrain the divergence between the policy before and after auxiliary updates to remain near a fixed KL-target of $\delta_{\text{clone}} = 0.01$.[1] The backbone's constraint-handling rule remains unchanged during the policy phase. The subsequent auxiliary update is not covered by the original feasibility guarantee of CPO (Achiam et al., 2017), and we do not claim theoretical cost feasibility for the complete phasic procedure. The resulting problem becomes

$$\min_\theta L^{\text{joint}} = L^{\text{aux_reward}} + \alpha_{\text{cost}} \cdot L^{\text{aux_cost}}, \text{ s.t. } \hat{\mathbb{E}}_t\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s_t) \,\|\, \pi_{\theta_\pi}(\cdot|s_t))\right] \leq \delta_{\text{clone}}. \quad (5)$$

This can be easily solved using dual ascent on the now implicit β_{clone} Lagrange multiplier. The advantage of representing the cloning penalty as a constraint is that it makes the regularization term invariant to changes in the scale of the auxiliary value and cost losses, which can vary substantially across environments and over the course of training. This is not entirely different from the KL-priors being transformed into constraints in e.g. Schulman et al. (2015; 2017). As a result, the auxiliary phase is governed by an interpretable policy-space constraint rather than by a static loss-weighting coefficient. We provide an empirical comparison in Section B, demonstrating how this dual formulation maintains stable policy divergence across diverse environments.

[1]The ideal value would be $\delta_{\text{clone}} = 0.0$ but we opt for a small constant for numerical stability and to ensure Slater's condition gives a sufficient constraint qualification in terms of π.

Algorithm 1 Phasic extension for on-policy constrained reinforcement learning.

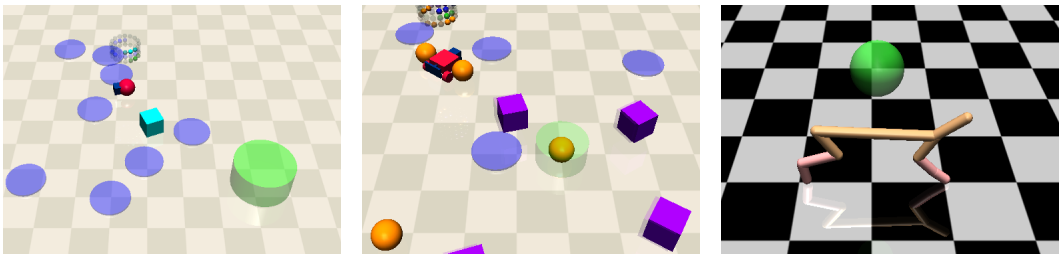
1: **for** phase $= 1, 2, \ldots$ **do**
2: Initialize empty buffer B
3: **for** iteration $= 1, 2, \ldots, N_\pi$ **do** ▷ Policy Phase
4: Perform rollouts under current policy π
5: Compute value targets $\hat{V}_t^{R,\text{targ}}$, $\hat{V}_t^{C,\text{targ}}$ for each state s_t
6: **for** epoch $= 1, 2, \ldots, E_\pi$ **do** ▷ Policy Epochs
7: UPDATEACTOR(π, rollout data) ▷ method-specific
8: **for** epoch $= 1, 2, \ldots, E_V$ **do** ▷ Critic Epochs
9: Optimize L_R^{value} w.r.t. θ_{V_R}
10: Optimize L_C^{value} w.r.t. θ_{V_C}
11: Add all $(s_t, \hat{V}_t^{R,\text{targ}}, \hat{V}_t^{C,\text{targ}})$ to B
12: Compute and store $\pi_{\theta_{\text{old}}}(\cdot \mid s_t)$ for all $s_t \in B$
13: **for** epoch $= 1, 2, \ldots, E_{\text{aux}}$ **do** ▷ Auxiliary Phase
14: Optimize L^{joint} w.r.t. θ_π, on all data in B
15: Optimize L_R^{value} w.r.t. θ_{V_R}, on all data in B
16: Optimize L_C^{value} w.r.t. θ_{V_C}, on all data in B

The generality of our scheme is demonstrated in Algorithm 1. During the policy phase, rollouts are collected under the current policy π and Generalized Advantage Estimation (Schulman et al., 2016) is applied to compute reward and cost value targets $\hat{V}_t^{R,\text{targ}}$ and $\hat{V}_t^{C,\text{targ}}$ for each visited state (lines 4–5). Following, the actor and critics are updated in individual loops. The actor is optimized for E_π epochs through the UPDATEACTOR call, which is specific to the underlying backbone CRL algorithm that we extend with this phasic framework. This leaves the constraint-handling logic entirely intact (lines 6–8). The two critics are then updated independently for E_V epochs, each minimizing a mean squared error loss against its respective value target (lines 9–11). Critically, rather than discarding the rollout data after each iteration, state observations and the corresponding computed value targets are stored in replay buffer B (line 12).

Disjoint Architecture. While a shared encoder would allow feature transfer implicitly, it also forces the policy gradient and value objectives to compete for the same parameters, a form of interference already observed in unconstrained phasic training (Cobbe et al., 2021; Aitchison & Sweetser, 2022). In constrained RL, the additional cost critic turns this into a three-way conflict between policy, reward-value, and cost-value objectives. Beyond gradient isolation, disjoint networks offer a second structural advantage in the phasic setting: each component can be updated on different data at different times without cross-contamination of their weight space. The critics are trained on buffered off-policy data during the auxiliary phase entirely independently of the policy network. We therefore adopt fully disjoint networks for the actor, the reward critic, and the cost critic (Figure 1), accepting the overhead of three-fold parameters and realizing the representational transfer through explicit distillation in the auxiliary phase (Cobbe et al., 2021).

4 EXPERIMENTS

We compare our universal phasic augmentation across the error-bound-based CPO (Achiam et al., 2017), the first-order Lagrangian CPPO-PID (Stooke et al., 2020), as well as the second-order Lagrangian approach TRPO-PID (Stooke et al., 2020). For each backbone, we compare its phasic extension with its unaugmented counterpart. Additional baselines include CUP (Yang et al., 2022), FOCOPS (Zhang et al., 2020), OnCRPO (Xu et al., 2021), PCPO (Yang et al., 2020), PDO (Achiam et al., 2017), PPOLag (Ray et al., 2019) and RCPO (Tessler et al., 2019). Following the established practice of evaluating learned representations through downstream task performance, as in SimCLR, DINO and LeJEPA (Chen et al., 2020; Caron et al., 2021; Balestriero & LeCun, 2025), we assess the practical benefit of auxiliary learning through performance on the CMDP itself, measured by achieved return and cost. Following the advice in Agarwal et al. (2021), we report median reward and median cost with 95 % bootstrap confidence intervals across 10 seeds. We use an expected



(a) `PointGoal1` (b) `CarButton1` (c) `HalfCheetahVelocity`

Figure 2: Representative environments used in our evaluation. `PointGoal1` is a constrained navigation task with static cost sources, whereas `CarButton1` adds a button-selection objective with multiple buttons, only one of which is active, and moving gremlins that generate costs upon collision. `HalfCheetahVelocity` contrasts with these object-centric navigation tasks by replacing external spatial constraints with a velocity constraint on an articulated locomotion agent.

episodic cost budget of $d = 25$, following the Safety Gym and Safety-Gymnasium benchmark protocols (Ray et al., 2019; Ji et al., 2023a). This conventional setting fixes the evaluation constraint consistently across methods.

4.1 Environments

To analyze the impact of augmenting the different methods with phasic updates, we evaluate our method on navigation and continuous control environments from the standardized Safety-Gymnasium (Ji et al., 2023a) environment, see Figure 2. The navigation environments, `CarGoal1`, `PointGoal1`, and `CarButton1`, expose object-centric external observations that require the agent to localize itself relative to goals and cost-inducing entities. `CarGoal1` and `PointGoal1` provide 48 such features, corresponding to goals, hazards, and vases. `CarButton1` increases this to 64 by adding button and gremlin observations, and further differs from the Goal tasks by introducing moving cost sources that generate costs upon collision. This makes `CarButton1` a stress test for constrained exploration, with prior Safety-Gymnasium benchmark results showing that common on-policy CRL baselines struggle to obtain high rewards while remaining within the prescribed cost budget (Ji et al., 2023a;b).

In contrast, `HalfCheetahVelocity` has no object-centric observation structure ($d_{\text{ext}} = 0$) but shifts the learning problem to velocity-constrained locomotion: an agent must coordinate an articulated body, and costs arise when locomotion velocity exceeds a threshold. A core difference in continuous control problems compared to navigation tasks is that the constraint in `HalfCheetahVelocity` is initially inactive as a randomly initialized policy typically fails to locomote, and hence rarely exceeds the velocity limit. The cost signal becomes informative after the agent has learned to move effectively, creating a qualitatively different reward-cost signal structure.

4.2 Phasic Augmentation Results

Figure 3 illustrates how phasic updates affect return and cost during training on `SafetyCarGoal1-v0`. We report medians and 95 % bootstrap confidence intervals across 10 seeds. CPPGPID reaches higher returns than CPPOPID early in training and retains this advantage throughout the plotted run. PTRPOPID also exceeds its TRPOPID backbone, with the return gap increasing later in training. These results suggest that auxiliary updates can extract further task-relevant value from the collected trajectories. The tables below assess whether these benefits extend to other tasks and backbones.

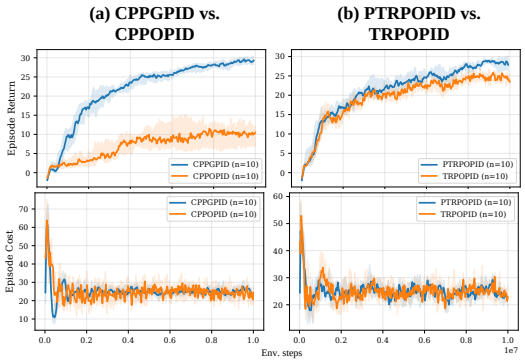


Figure 3: Phasic vs. non-phasic training curves on `SafetyCarGoal1-v0`.

Table 1: Performance on `CarGoal1` and `PointGoal1` after 10^6 training steps. We report median with 95 % bootstrap confidence intervals across seeds. († denotes our phasic extension). Phasic variants use $\alpha_{\text{cost}} = 0.3$.

	CarGoal1		**PointGoal1**	
Algorithm	R (↑)	C (↓)	R (↑)	C (↓)
PhasicCPO†	27.0 (26.0, 27.9)	25.4 (24.5, 28.3)	23.7 (23.6, 24.3)	25.7 (24.2, 27.6)
CPO	23.5 (22.3, 25.0)	28.2 (26.6, 29.2)	21.4 (19.8, 22.2)	28.2 (27.3, 29.3)
CPPGPID†	**29.0 (28.5, 30.1)**	26.0 (24.3, 26.6)	24.3 (23.8, 24.8)	24.3 (23.4, 25.2)
CPPOPID	10.5 (6.2, 12.3)	22.6 (21.1, 24.8)	10.2 (6.5, 13.3)	23.2 (19.5, 26.8)
PTRPOPID†	28.2 (26.0, 30.1)	23.7 (20.4, 26.0)	**24.5 (23.6, 24.8)**	25.5 (23.4, 27.3)
TRPOPID	24.5 (22.7, 25.6)	23.9 (22.1, 24.7)	20.4 (18.8, 21.9)	25.9 (22.7, 28.2)
CUP	9.3 (3.1, 17.7)	27.1 (23.0, 33.3)	14.9 (12.8, 16.9)	27.7 (24.4, 30.5)
FOCOPS	19.1 (16.5, 21.2)	29.1 (26.7, 32.1)	13.8 (11.0, 17.8)	30.1 (25.7, 34.1)
OnCRPO	24.6 (23.5, 26.1)	30.6 (28.5, 33.2)	21.6 (20.4, 22.5)	30.4 (27.3, 32.1)
PCPO	19.6 (17.3, 20.9)	32.2 (29.1, 34.3)	17.9 (17.2, 19.8)	30.1 (28.1, 33.0)
PDO	1.1 (-0.6, 2.4)	26.8 (24.9, 41.8)	0.6 (0.1, 2.2)	29.5 (14.9, 50.1)
PPOLag	15.4 (12.8, 17.1)	23.0 (17.7, 24.2)	11.8 (9.0, 13.0)	23.5 (22.2, 27.3)
RCPO	24.3 (22.6, 25.5)	25.7 (22.9, 27.0)	18.1 (14.7, 20.2)	22.1 (21.2, 29.2)

Table 1 presents results for the `CarGoal1` and `PointGoal1` navigation task environments with object-centric external observations. In this regime, across all backbone algorithms, extending policy optimization with our phasic framework yields higher returns, while not substantially increasing the cost footprint. This indicates that the information that is sampled from the environment still contains exploitable information, when the backbone algorithm discards it, while our phasic extension leverages it to achieve a higher overall result. In particular, phasic-extended algorithms yield the highest returns for both environments while still remaining within or close to the cost budget.

In particular, in the case of CPPOPID (Stooke et al., 2020), the phasic extension adds a substantial reward gain of $+18.5$, while median cost rises from 22.6 to 26.0, slightly above the prescribed budget. Although the Point and Car robots differ in their actuation dynamics, the phasic extensions improve upon their respective backbones. This suggests that the gains are not tied to a particular robot but the benefit of phasic representation refinement generalizes across both environments.

Table 2 presents the results for `CarButton1`, our stress test for constrained exploration. In this environment, the phasic extension improves the episodic return of both Lagrangian-based backbones

Table 2: Performance on `CarButton1` and `HalfCheetahVelocity` after 10^6 training steps. We report median with 95 % bootstrap confidence intervals across seeds. († is our phasic extension). Phasic variants use $\alpha_{\text{cost}} = 0.3$.

	CarButton1		**HalfCheetahVelocity**	
Algorithm	R (↑)	C (↓)	R (↑)	C (↓)
PhasicCPO†	1.0 (0.5, 1.3)	33.9 (29.6, 35.7)	2081.4 (1994.5, 2736.7)	19.9 (13.4, 20.9)
CPO	1.0 (0.8, 1.4)	35.1 (32.5, 38.5)	2605.2 (1855.4, 2755.6)	21.2 (20.3, 23.1)
CPPGPID†	0.6 (0.2, 0.9)	25.2 (23.0, 27.7)	2504.2 (2272.5, 2589.9)	22.2 (19.7, 30.5)
CPPOPID	-0.8 (-1.2, -0.5)	25.3 (19.0, 33.8)	3029.3 (3006.8, 3047.6)	25.6 (23.9, 26.9)
PTRPOPID†	0.7 (0.3, 1.3)	24.4 (22.5, 26.9)	2072.0 (2019.1, 2853.8)	20.5 (17.5, 27.8)
TRPOPID	0.2 (0.1, 0.6)	21.7 (20.1, 29.0)	2266.8 (1861.2, 2833.7)	23.9 (21.3, 25.2)
CUP	**3.1 (1.3, 4.4)**	163.5 (105.9, 194.3)	2508.3 (1855.3, 2823.0)	16.7 (10.1, 31.7)
FOCOPS	1.6 (1.0, 1.7)	39.9 (33.9, 54.0)	2898.6 (2789.1, 3023.0)	41.9 (32.1, 49.7)
OnCRPO	1.3 (0.8, 1.5)	36.6 (30.4, 41.4)	2296.3 (1798.9, 2804.7)	24.2 (22.3, 25.1)
PCPO	0.1 (-0.1, 0.4)	39.5 (34.7, 42.1)	1458.3 (1367.1, 1647.3)	19.7 (17.9, 21.4)
PDO	-0.0 (-1.0, 0.5)	39.8 (22.6, 61.8)	2847.8 (2786.2, 2873.7)	25.7 (22.4, 27.8)
PPOLag	-0.1 (-0.4, 0.2)	32.9 (28.5, 37.0)	**3034.2 (3011.5, 3045.7)**	22.7 (20.2, 24.9)
RCPO	0.9 (0.5, 1.2)	21.2 (17.3, 28.1)	2324.4 (1802.7, 2806.5)	20.1 (17.3, 26.4)

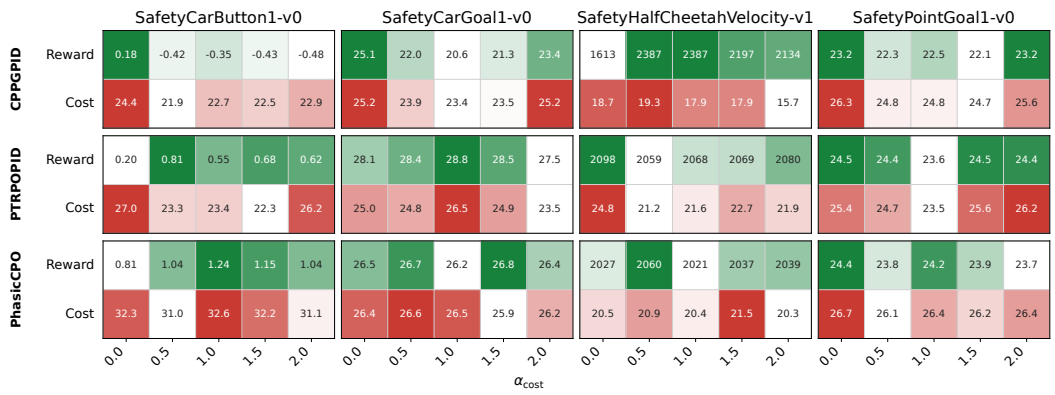


Figure 4: Ablation over α_{cost} across environments and algorithms (upper row: episode return; lower row: episode cost). Green indicates higher return, while red indicates higher cost.

without meaningfully degrading constraint satisfaction. The median return of TRPO-PID increases from 0.2 to 0.7, corresponding to a 3.5-fold improvement, while CPPO-PID shifts from a negative median return of -0.8 to a positive median return of 0.6.

The pattern reverses for `HalfCheetahVelocity` – across all three backbones, the phasic extension reduces episodic return. This contrast is consistent with the task characteristics discussed in Section 4.1. Unlike the navigation environments, `HalfCheetahVelocity` provides no object-centric spatial description of the environment and therefore exposes a substantially lighter observation space. Its constraint is instead determined by locomotion velocity and typically becomes active only after the policy has already learned to move effectively. The auxiliary phase may therefore provide comparatively little additional representational value in this setting. Instead, repeated distillation updates may perturb features that already support coordinated locomotion.

To complement these aggregate results, Section E reports the return and cost distributions obtained from 500 evaluation rollouts per seed for every phasic method and its corresponding backbone across all environments.

4.3 Sensitivity to the Cost-Distillation Weight

As shown in Figure 4, most environment–backbone combinations, performance varies only moderately with α_{cost}, and no single value consistently dominates across tasks. The most notable deviations occur for CPPGPID. On `CarButton1`, the strongest result is obtained with $\alpha_{\text{cost}} = 0$, whereas on `HalfCheetahVelocity`, an intermediate weighting performs better. This variation indicates that the optimal balance between reward- and cost-value distillation is task-dependent rather than governed by a universal setting.

More broadly, the relative robustness of the phasic variants across a wide range of α_{cost} values suggests that their benefit does not primarily depend on finely calibrating the cost-value objective. Instead, the auxiliary phase appears to benefit from providing the policy encoder with an additional learning signal that consistently relates visited states to the dynamics encountered during training. Detailed numerical results, including the interquartile range across seeds, are reported in Section C. The corresponding return and cost learning curves over the full training duration are provided in Section D.

5 Limitations

Our results suggest that phasic training can improve data utilization and representation learning in on-policy constrained reinforcement learning, but several limitations remain.

Our experiments are consistent with the hypothesis that auxiliary reward- and cost-value distillation improves feature quality, but they do not isolate or directly investigate the underlying representational mechanism. In particular, we do not test whether the learned latent space captures transition

structure, cost boundaries, or successor-like state geometry. Future work should therefore analyze whether phasic CRL improves representations in a way that is explicitly aligned with environment dynamics rather than merely improving value prediction.

Alternative forms of feature distillation. We only evaluate one particular form of feature distillation. In our framework, reward- and cost-related information is transferred to the policy network through auxiliary value heads trained by regression. This design follows the general spirit of phasic actor–critic methods, but it should not be interpreted as exhausting the possible ways of transferring critic information into the policy. Alternative distillation mechanisms could align intermediate feature representations, use advantage- rather than value-based targets or introduce dynamics-aware successor state auxiliary objectives.

Misaligned reward and cost distillation needs. Another limitation is that our auxiliary phase distills reward- and cost-value information within a single joint update. This may be overly restrictive. Prior work on dual-network actor–critic learning shows that policy and value objectives can differ substantially in gradient noise and in their preferred bias–variance trade-offs for return estimation (Aitchison & Sweetser, 2022). In constrained RL, this misalignment may extend to the reward and cost critics themselves, whose targets can differ in scale, sparsity, temporal structure, and relevance across the state space. A single auxiliary loop with one buffer, one update schedule, one KL budget, and one cost-weighting coefficient may therefore not provide the right learning conditions for both signals.

This concern is reinforced by recent analyses of phasic policy-gradient methods, which identify policy regularization and data diversity as central factors behind successful feature distillation (Wang et al., 2023). In CRL, the relevant data diversity for reward learning may differ from that for cost learning: reward prediction may benefit from broad task-progress coverage, whereas cost prediction may depend disproportionately on rare constraint-relevant transitions. Future work could therefore study asymmetric phasic designs in which reward and cost distillation use separate buffers, target estimators, update frequencies, or regularization strengths.

6 Conclusion

In this work, we propose a modular phasic extension that reuses collected trajectories for auxiliary reward and cost learning in on-policy constrained reinforcement learning.

Across the two Goal tasks, all three phasic variants achieve higher median return than their respective backbones, and four of the six comparisons also yield lower median cost. Results on `CarButton1` depend on the backbone, while all three variants obtain lower returns on `HalfCheetahVelocity`.

Overall, these results suggest that phasic CRL is best understood as a representation- and data-utilization mechanism whose usefulness depends on whether the environment provides reusable structure beyond the immediate on-policy update. More broadly, our results point to representation quality and data utilization as a lever in constrained policy optimization that can substantially boost the performance of the underlying algorithm.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. *arXiv preprint arXiv:1705.10528*, 2017.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html`.

Matthew Aitchison and Penny Sweetser. DNA: Proximal policy optimization with a dual network architecture. *arXiv preprint arXiv:2206.10027*, 2022.

Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, Boca Raton, 1999.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. URL https://arxiv.org/abs/2511.08544.

Guillaume Carlier, Enis Chenchene, and Katharina Eichinger. Wasserstein medians: Robustness, PDE characterization, and numerics. *SIAM Journal on Mathematical Analysis*, 56(5):6483–6520, 2024. doi: 10.1137/23M1624786.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021. URL https://arxiv.org/abs/2104.14294.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Karl W. Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2020–2027. PMLR, 2021. URL https://proceedings.mlr.press/v139/cobbe21a.html.

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

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330v1*, 2022. URL https://arxiv.org/abs/2205.10330v1.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. In *Advances in Neural Information Processing Systems*, 2023a.

Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. OmniSafe: An infrastructure for accelerating safe reinforcement learning research. *arXiv preprint arXiv:2305.09304*, 2023b.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. Technical report, OpenAI, 2019. URL https://cdn.openai.com/safexp-short.pdf.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

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

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

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID lagrangian methods. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkfrvsA9FX.

Kaixin Wang, Daquan Zhou, Jiashi Feng, and Shie Mannor. PPG reloaded: An empirical study on what matters in phasic policy gradient. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 36694–36713. PMLR, 2023. URL https://proceedings.mlr.press/v202/wang23aw.html.

Tengyu Xu, Yingbin Liang, and Guanghui Lan. CRPO: A new approach for safe reinforcement learning with convergence guarantee. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11480–11491, 2021. URL https://proceedings.mlr.press/v139/xu21a.html.

Long Yang, Jiaming Ji, Juntao Dai, Linrui Zhang, Binbin Zhou, Pengfei Li, Yaodong Yang, and Gang Pan. Constrained update projection approach to safe policy optimization. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9111–9124, 2022.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rke3TJrtPS.

Kisung You, Dennis Shung, and Mauro Giuffrè. On the Wasserstein median of probability measures. *Journal of Computational and Graphical Statistics*, 34(1):253–266, 2025.

Yiming Zhang, Quan Vuong, and Keith Ross. First order constrained optimization in policy space. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15338–15349, 2020.

A IMPLEMENTATION DETAILS.

Unless stated otherwise, we use the following hyperparameters across all experiments.

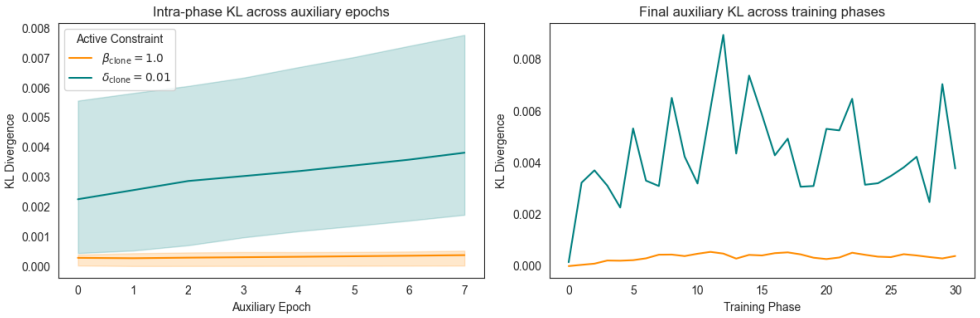
$$\begin{aligned} N_\pi &= 16 & E_\pi &= 1 \\ E_V &= 6 & E_{\text{aux}} &= 6 \\ M_{\text{aux}} &= 10 & \alpha_{\text{cost}} &= 0.3 \\ \delta_{\text{clone}} &= 0.01 & d &= 25 \end{aligned}$$

Each policy iteration collects $20{,}000$ environment transitions. The actor and both critics use two hidden layers of 256 units with tanh activations and learning rates of 3×10^{-4}. Reward and cost targets use discount factors of 0.99 and GAE parameters of 0.95. Observations are normalized, while rewards and costs are not.

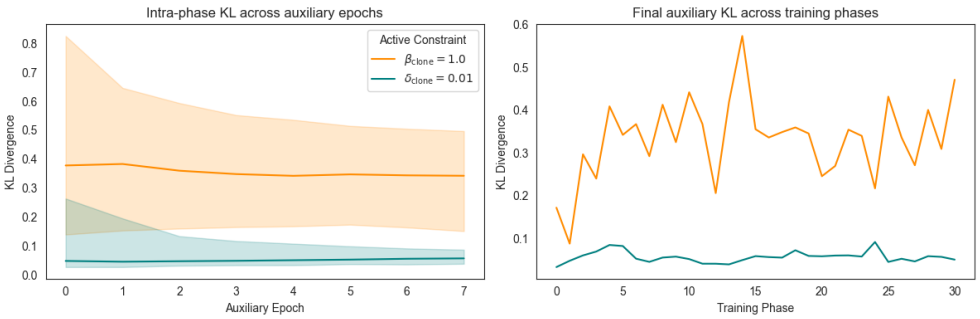
The policy-phase minibatch size is 64. The auxiliary minibatch size is $\lfloor |B|/(N_\pi M_{\text{aux}}) \rfloor$, where M_{aux} specifies the number of auxiliary minibatches per retained rollout batch. With a full buffer, this gives $2{,}000$ samples per minibatch and 160 minibatches per auxiliary epoch.

B IMPACT OF CONSTRAINT-BASED PHASIC PENALTY

As illustrated in Figure 5, our adaptive formulation maintains a highly consistent level of policy-space divergence across environments, whereas a fixed coefficient produces markedly different KL values. This instability in the fixed approach arises because the relative influence of a static cloning loss heavily depends on the scale of the auxiliary regression objectives, which inherently vary across tasks and throughout training. By directly constraining policy divergence, our adaptive approach provides a more interpretable and transferable mechanism for regularizing auxiliary updates, effectively alleviating the need to fine-tune a static hyperparameter for each new environment.



(a) `CarGoal1`



(b) `HalfCheetahVelocity`

Figure 5: Comparison of a fixed cloning coefficient ($\beta_{\text{clone}} = 1$) and the proposed adaptive KL-constrained formulation with target divergence $\delta_{\text{clone}} = 0.01$ in two representative environments. The reported KL divergence is measured between the policy obtained after the policy phase and the policy resulting from auxiliary feature distillation. Left: KL divergence accumulated across auxiliary epochs, aggregated over all training phases. Right: KL divergence after the final auxiliary epoch of each training phase.

C Detailed Alpha Cost Ablation Results

Supplementing the aggregated performance heatmap in Section 4.3, Tables 3 through 6 report the exact median return and cost values alongside the interquartile range (IQR) to illustrate cross-seed dispersion. While the main text highlights the overall trend that intermediate cost-distillation weights generally yield stable median returns, the inclusion of the IQR here provides a clearer picture of the variance across different seeds and environments. Notably, this dispersion data demonstrates that the phasic variants remain relatively robust across a wide range of weights without suffering from extreme performance collapse or severe constraint violations.

Table 3: Final performance on SafetyCarButton1-v0. Sweep entries report median ± IQR; baseline entries report the median with the reported interval.

Algorithm	$\alpha_{\text{cost}} = 0.0$	$\alpha_{\text{cost}} = 0.5$	$\alpha_{\text{cost}} = 1.0$	$\alpha_{\text{cost}} = 1.5$	$\alpha_{\text{cost}} = 2.0$	Baseline
CPPGPID	Ret: **0.184** ± 0.802 Cost: 24.418 ± 3.359	Ret: -0.424 ± 0.275 Cost: **21.85** ± 4.961	Ret: -0.35 ± 0.353 Cost: 22.675 ± 3.511	Ret: -0.43 ± 0.283 Cost: 22.473 ± 1.078	Ret: -0.479 ± 0.245 Cost: 22.89 ± 3.544	Ret: -0.8 (-1.2, -0.5) Cost: 25.3 (19.0, 33.8)
PTRPOPID	Ret: 0.204 ± 0.696 Cost: 26.952 ± 5.341	Ret: **0.805** ± 0.482 Cost: 23.29 ± 2.465	Ret: 0.547 ± 0.632 Cost: 23.387 ± 5.866	Ret: 0.675 ± 0.66 Cost: 22.265 ± 5.054	Ret: 0.616 ± 0.329 Cost: 26.215 ± 9.535	Ret: 0.2 (0.1, 0.6) Cost: **21.7** (20.1, 29.0)
PhasicCPO	Ret: 0.809 ± 0.605 Cost: 32.345 ± 4.585	Ret: 1.038 ± 0.796 Cost: **30.988** ± 2.333	Ret: **1.238** ± 0.496 Cost: 32.628 ± 0.867	Ret: 1.152 ± 0.458 Cost: 32.183 ± 2.052	Ret: 1.045 ± 0.472 Cost: 31.11 ± 5.399	Ret: 1.0 (0.8, 1.4) Cost: 35.1 (32.5, 38.5)

Table 4: Final performance on SafetyCarGoal1-v0. Sweep entries report median ± IQR; baseline entries report the median with the reported interval.

Algorithm	$\alpha_{\text{cost}} = 0.0$	$\alpha_{\text{cost}} = 0.5$	$\alpha_{\text{cost}} = 1.0$	$\alpha_{\text{cost}} = 1.5$	$\alpha_{\text{cost}} = 2.0$	Baseline
CPPGPID	Ret: **25.135** ± 2.346 Cost: 25.23 ± 3.794	Ret: 21.995 ± 2.159 Cost: 23.872 ± 1.241	Ret: 20.615 ± 2.868 Cost: 23.427 ± 2.014	Ret: 21.318 ± 3.509 Cost: 23.465 ± 1.636	Ret: 23.394 ± 2.447 Cost: 25.188 ± 1.253	Ret: 10.5 (6.2, 12.3) Cost: **22.6** (21.1, 24.8)
PTRPOPID	Ret: 28.064 ± 2.75 Cost: 25.015 ± 2.728	Ret: 28.355 ± 2.069 Cost: 24.75 ± 4.74	Ret: **28.841** ± 0.973 Cost: 26.465 ± 2.894	Ret: 28.46 ± 1.429 Cost: 24.92 ± 2.396	Ret: 27.525 ± 1.807 Cost: **23.487** ± 2.048	Ret: 24.5 (22.7, 25.6) Cost: 23.9 (22.1, 24.7)
PhasicCPO	Ret: 26.523 ± 1.186 Cost: 26.42 ± 0.765	Ret: 26.743 ± 1.683 Cost: 26.562 ± 1.593	Ret: 26.167 ± 1.275 Cost: 26.457 ± 0.6	Ret: **26.849** ± 1.662 Cost: **25.883** ± 1.536	Ret: 26.418 ± 1.721 Cost: 26.158 ± 0.594	Ret: 23.5 (22.3, 25.0) Cost: 28.2 (26.6, 29.2)

Table 5: Final performance on SafetyHalfCheetahVelocity-v1. Sweep entries report median ± IQR; baseline entries report the median with the reported interval.

Algorithm	$\alpha_{\text{cost}} = 0.0$	$\alpha_{\text{cost}} = 0.5$	$\alpha_{\text{cost}} = 1.0$	$\alpha_{\text{cost}} = 1.5$	$\alpha_{\text{cost}} = 2.0$	Baseline
CPPGPID	Ret: 1612.53 ± 686.279 Cost: 18.653 ± 6.186	Ret: 2387.36 ± 340.765 Cost: 19.293 ± 3.974	Ret: 2386.77 ± 523.112 Cost: 17.933 ± 6.822	Ret: 2196.93 ± 729.574 Cost: 17.885 ± 3.599	Ret: 2134.38 ± 901.319 Cost: **15.67** ± 12.324	Ret: **3029.3** (3006.8, 3047.6) Cost: 25.6 (23.9, 26.9)
PTRPOPID	Ret: 2098.37 ± 544.505 Cost: 24.825 ± 2.822	Ret: 2058.52 ± 591.439 Cost: **21.17** ± 4.086	Ret: 2068.01 ± 523.554 Cost: 21.578 ± 3.454	Ret: 2069.03 ± 104.026 Cost: 22.678 ± 4.852	Ret: 2080.38 ± 501.071 Cost: 21.938 ± 4.866	Ret: **2266.8** (1861.2, 2833.7) Cost: 23.9 (21.3, 25.2)
PhasicCPO	Ret: 2027.2 ± 599.578 Cost: 20.54 ± 2.934	Ret: 2059.61 ± 784.258 Cost: 20.9 ± 1.281	Ret: 2021.42 ± 546.159 Cost: 20.423 ± 3.081	Ret: 2036.8 ± 502.13 Cost: 21.537 ± 4.435	Ret: 2038.79 ± 694.198 Cost: **20.308** ± 3.95	Ret: **2605.2** (1855.4, 2755.6) Cost: 21.2 (20.3, 23.1)

Table 6: Final performance on SafetyPointGoal1-v0. Sweep entries report median ± IQR; baseline entries report the median with the reported interval.

Algorithm	$\alpha_{\text{cost}} = 0.0$	$\alpha_{\text{cost}} = 0.5$	$\alpha_{\text{cost}} = 1.0$	$\alpha_{\text{cost}} = 1.5$	$\alpha_{\text{cost}} = 2.0$	Baseline
CPPGPID	Ret: **23.172** ± 1.551 Cost: 26.338 ± 1.65	Ret: 22.341 ± 2.892 Cost: 24.803 ± 1.564	Ret: 22.479 ± 2.583 Cost: 24.793 ± 3.483	Ret: 22.138 ± 1.859 Cost: 24.672 ± 4.261	Ret: 23.152 ± 1.654 Cost: 25.562 ± 1.459	Ret: 10.2 (6.5, 13.3) Cost: **23.2** (19.5, 26.8)
PTRPOPID	Ret: **24.544** ± 2.22 Cost: 25.35 ± 1.382	Ret: 24.37 ± 1.394 Cost: 24.673 ± 2.038	Ret: 23.616 ± 1.383 Cost: **23.468** ± 2.159	Ret: 24.48 ± 1.292 Cost: 25.552 ± 2.441	Ret: 24.441 ± 2.173 Cost: 26.238 ± 1.306	Ret: 20.4 (18.8, 21.9) Cost: 25.9 (22.7, 28.2)
PhasicCPO	Ret: **24.353** ± 1.68 Cost: 26.715 ± 0.869	Ret: 23.803 ± 1.192 Cost: **26.125** ± 0.62	Ret: 24.241 ± 0.658 Cost: 26.375 ± 1.454	Ret: 23.895 ± 1.387 Cost: 26.25 ± 0.52	Ret: 23.681 ± 0.959 Cost: 26.417 ± 1.114	Ret: 21.4 (19.8, 22.2) Cost: 28.2 (27.3, 29.3)

D ALPHA COST ABLATION TRAINING CURVES

While the heatmaps and tabular data in the preceding sections capture the final converged performance of the policies, they inherently obscure the temporal dynamics of the learning process. The learning curves presented in this section illustrate the evolution of both episode return and episode cost across the full training duration. Analyzing these trajectories is crucial, as varying the cost-distillation weight (α_{cost}) can yield similar asymptotic final performance while exhibiting markedly different transient behaviors.

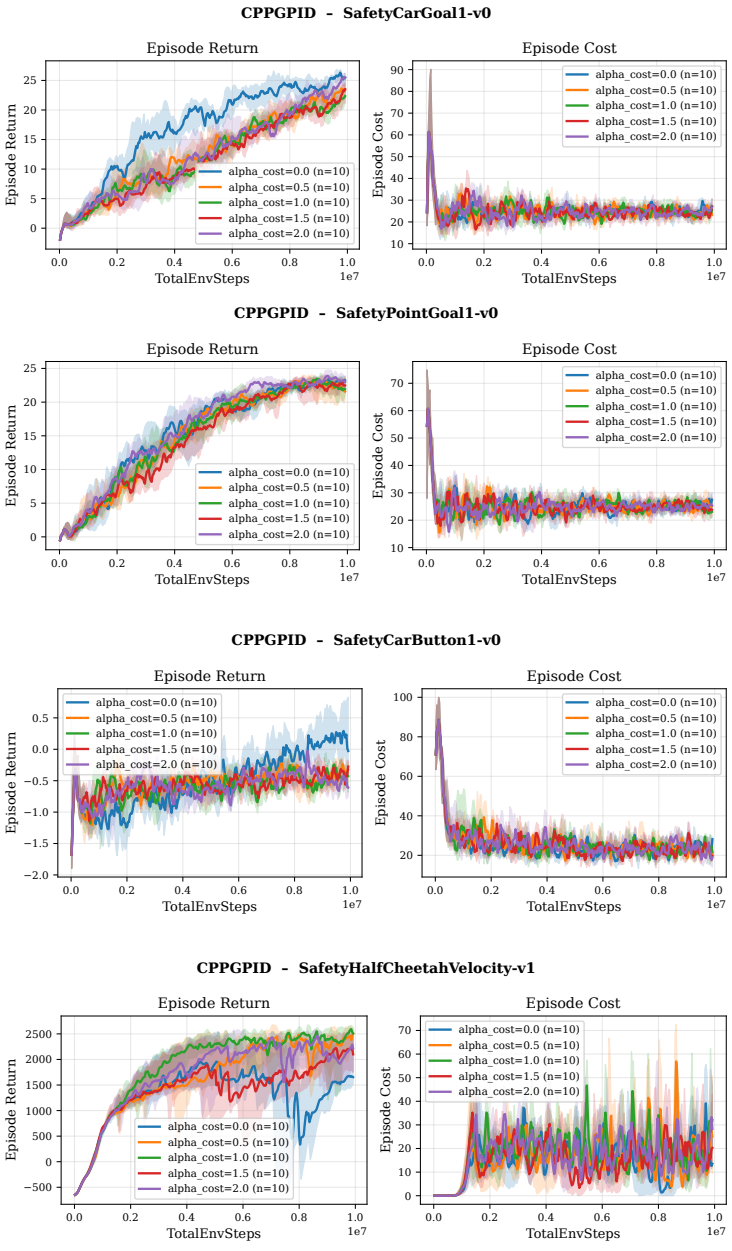


Figure 6: **CPPGPID Training Dynamics:** Learning curves detailing the effect of varying α_{cost} across all evaluated environments. Solid lines denote the median, while shaded regions represent the interquartile range across 10 seeds.

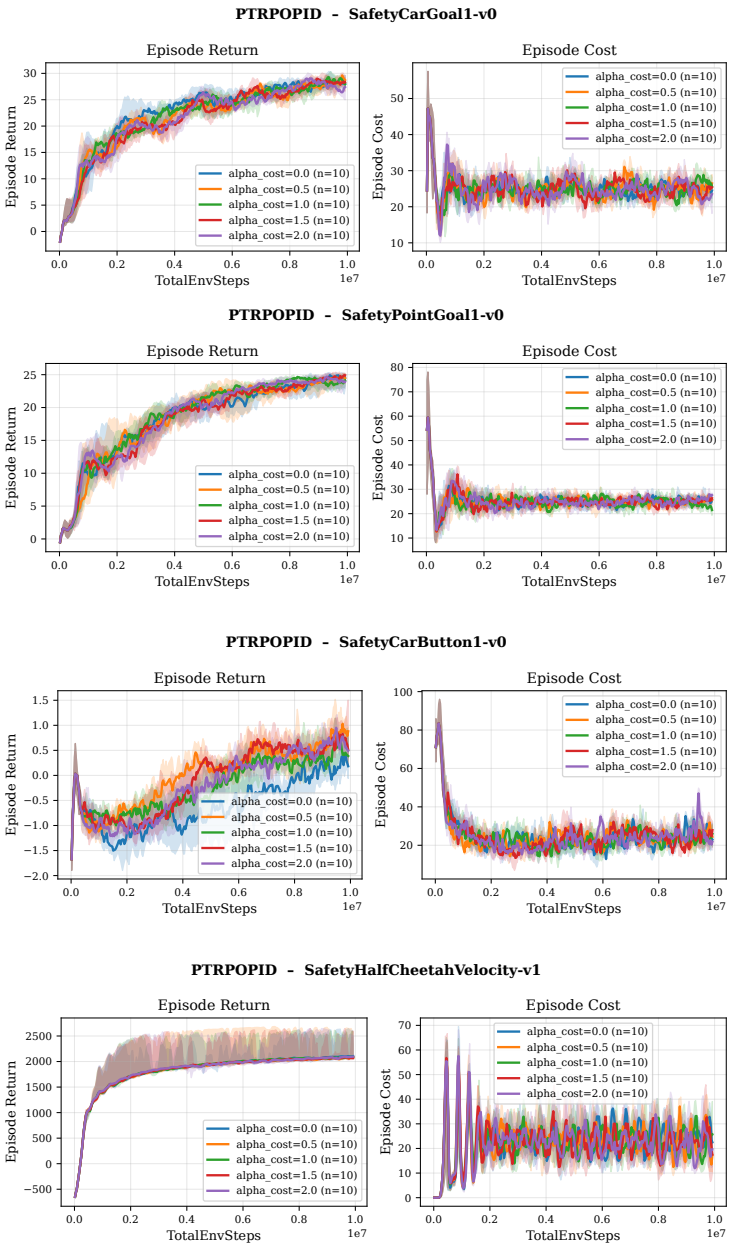


Figure 7: **PTRPOPID Training Dynamics:** Learning curves detailing the effect of varying α_{cost} across all evaluated environments. Solid lines denote the median, while shaded regions represent the interquartile range across 10 seeds.

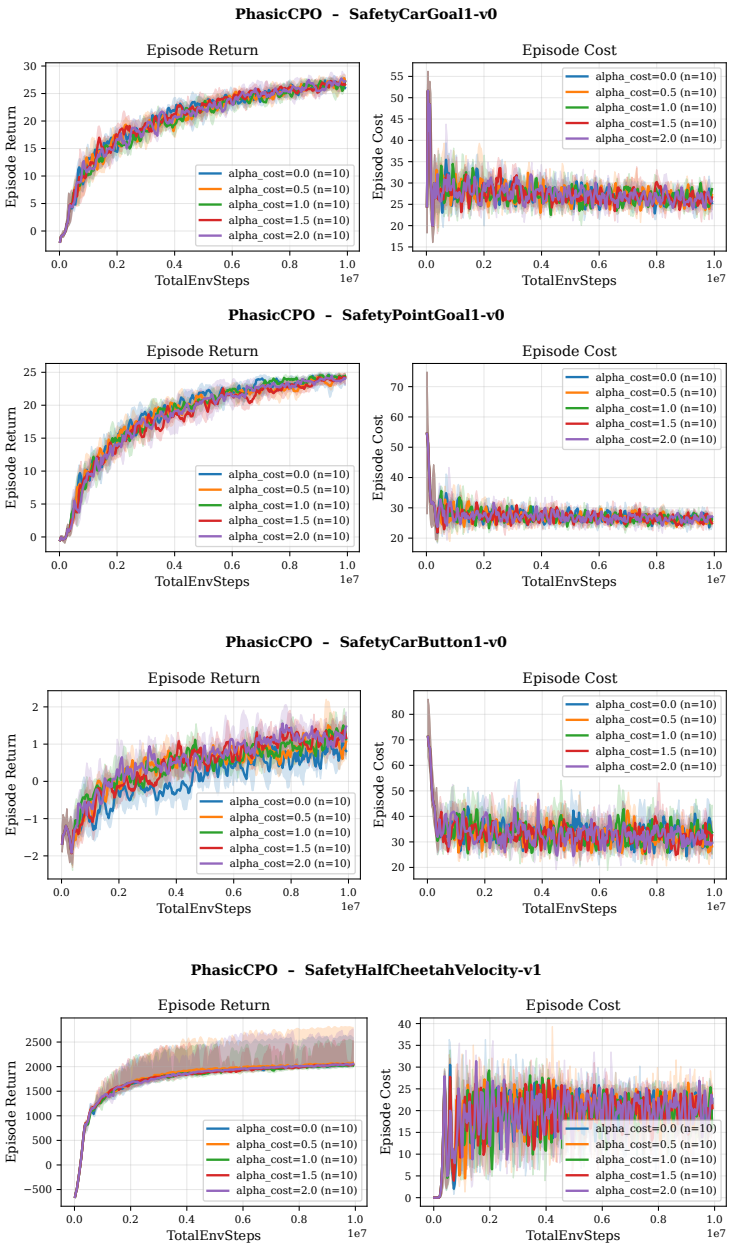


Figure 8: **PhasicCPO Training Dynamics:** Learning curves detailing the effect of varying α_{cost} across all evaluated environments. Solid lines denote the median, while shaded regions represent the interquartile range across 10 seeds.

E DISTRIBUTIONS OVER CONVERGED POLICY ROLLOUTS

In addition to the mean and bootstrap confidence intervals reported in the main text, we investigate the distributional behavior of the algorithms after training. This analysis is particularly important for Constrained Markov Decision Processes (CMDPs), which are inherently underconstrained: a cost budget constraint, $\mathbb{E}_\pi[C(a,s)] \leq d$, can be satisfied through entirely different behaviors. For instance, a policy might consistently incur costs just below the threshold (low variance), or it might behave highly erratically, occasionally incurring massive costs while remaining safe on average (high variance). Both yield the same expected value, but they represent vastly different operational realities during deployment.

To capture these dynamics, we deploy the converged policies at the end of training and generate 500 rollouts per seed. To robustly aggregate these independent and identically distributed (i.i.d.) histograms, we compute the Wasserstein median over the different seeds (You et al., 2025). The Wasserstein median serves as a robust estimator of the Wasserstein barycenter (Carlier et al., 2024), making it an ideal aggregation mechanism for distributional rollout data.

The following figures group these distributions by environment. To facilitate a direct visual comparison of how auxiliary distillation alters the shape of the return and cost distributions, each phasic algorithm (top) is paired directly with its vanilla baseline counterpart (bottom).

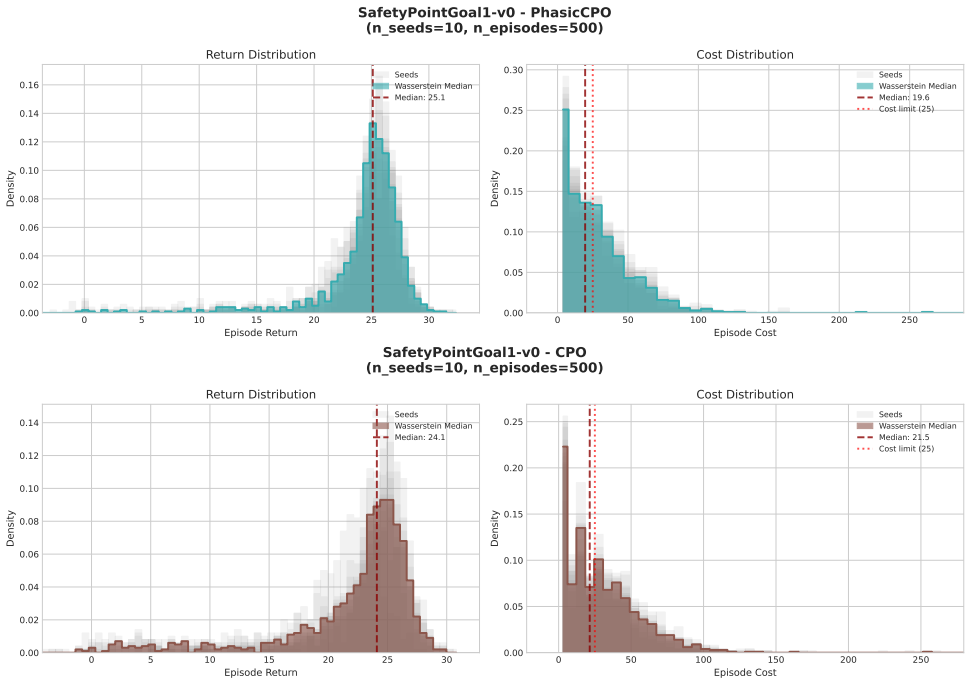


Figure 9: Deployment rollout distributions on `SafetyPointGoal1-v0` after 10^6 training steps. The phasic extension (PhasicCPO) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPO) on the bottom.

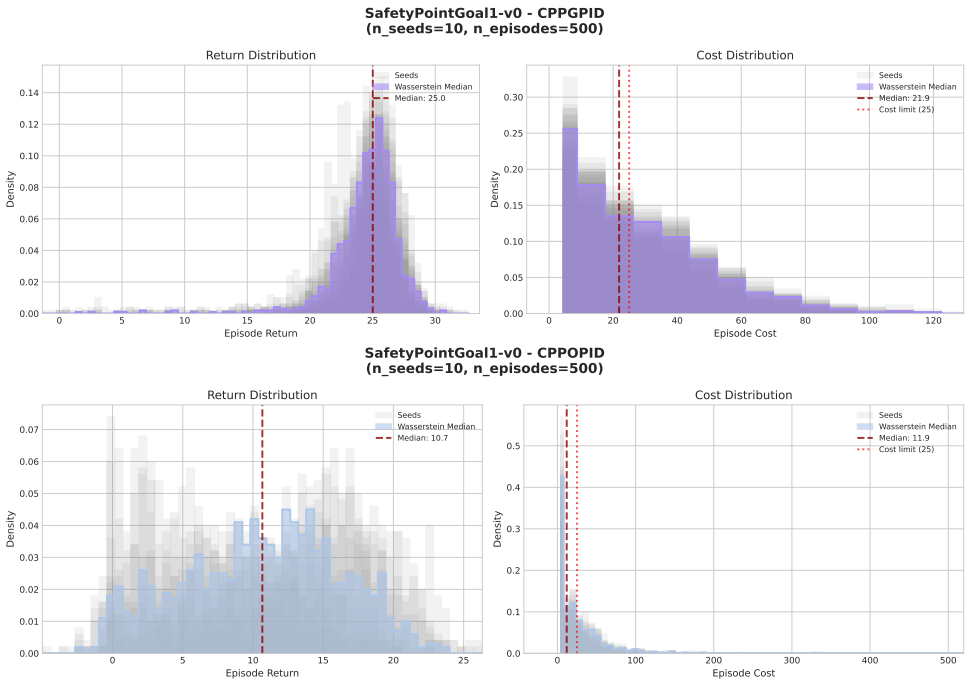


Figure 10: Deployment rollout distributions on `SafetyPointGoal1-v0` after 10^6 training steps. The phasic extension (CPPGPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPPOPID) on the bottom.

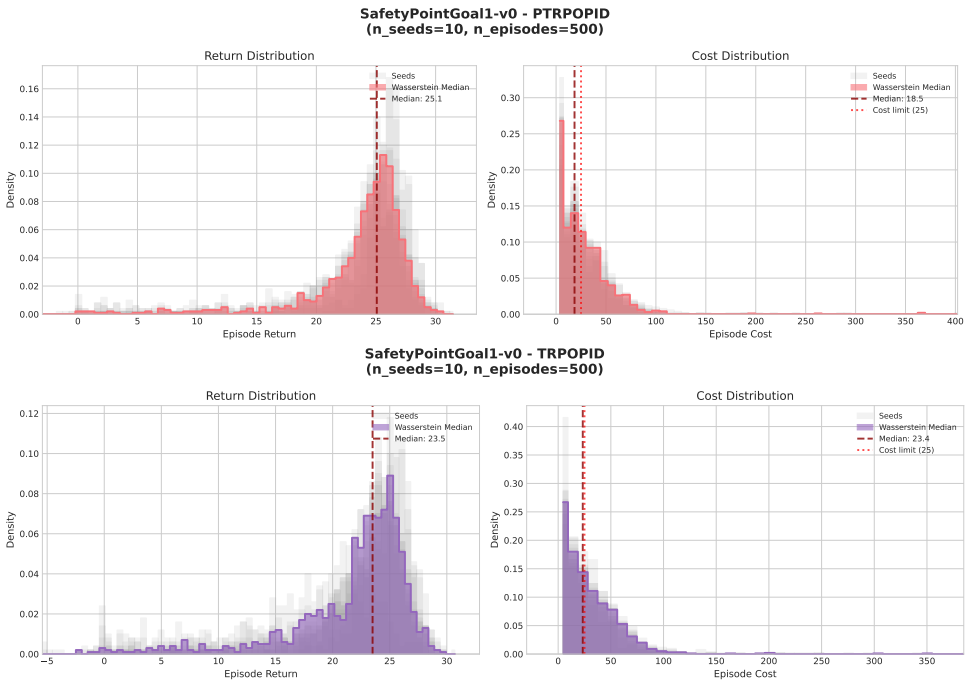


Figure 11: Deployment rollout distributions on `SafetyPointGoal1-v0` after 10^6 training steps. The phasic extension (PTRPOPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (TRPOPID) on the bottom.

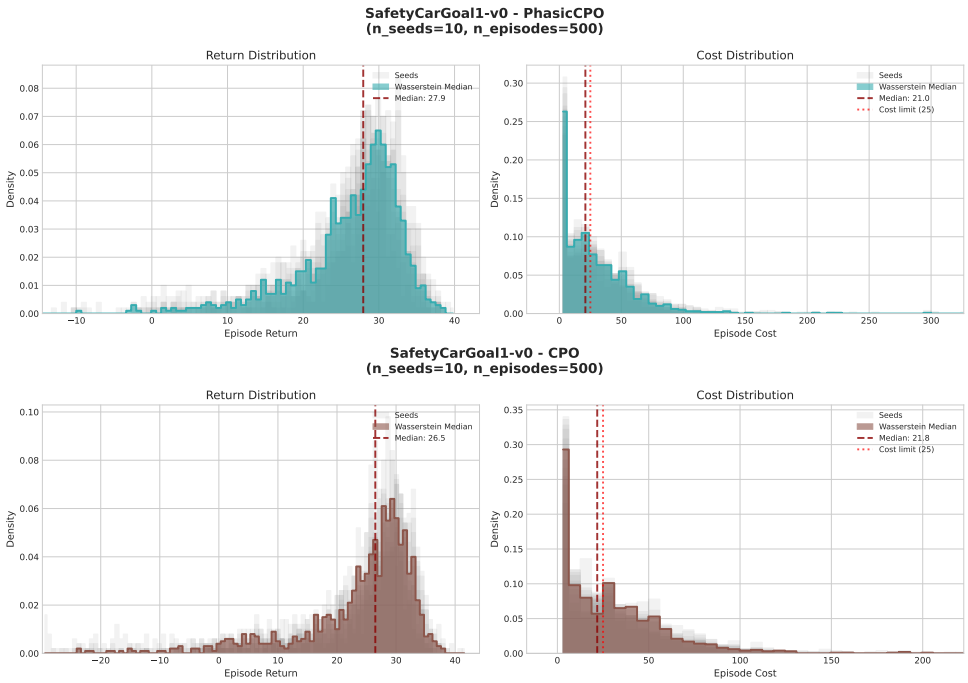


Figure 12: Deployment rollout distributions on `SafetyCarGoal1-v0` after 10^6 training steps. The phasic extension (PhasicCPO) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPO) on the bottom.

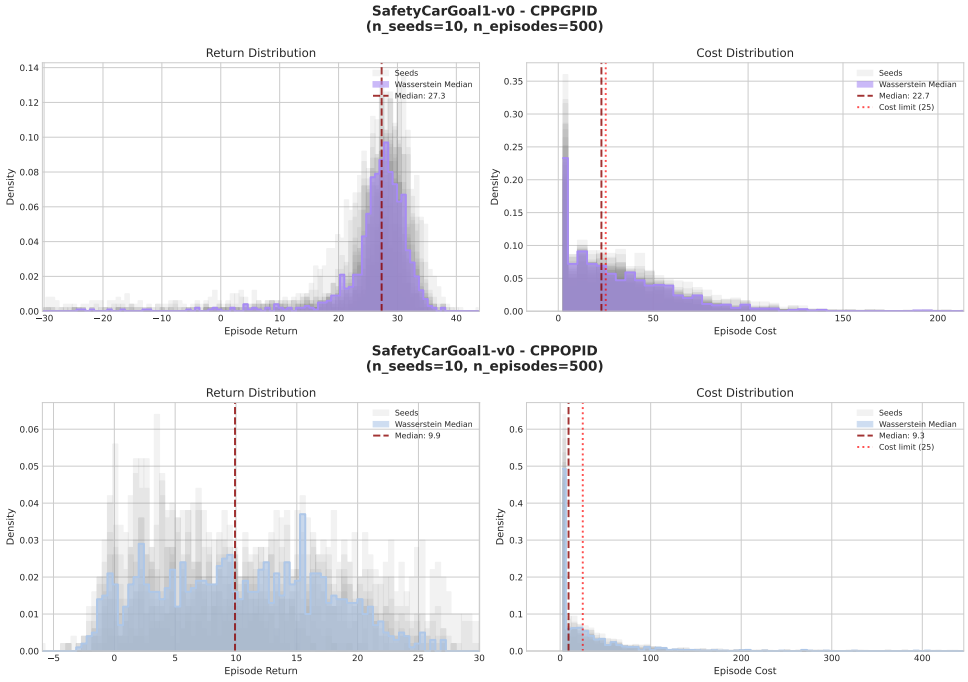


Figure 13: Deployment rollout distributions on `SafetyCarGoal1-v0` after 10^6 training steps. The phasic extension (CPPGPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPPOPID) on the bottom.

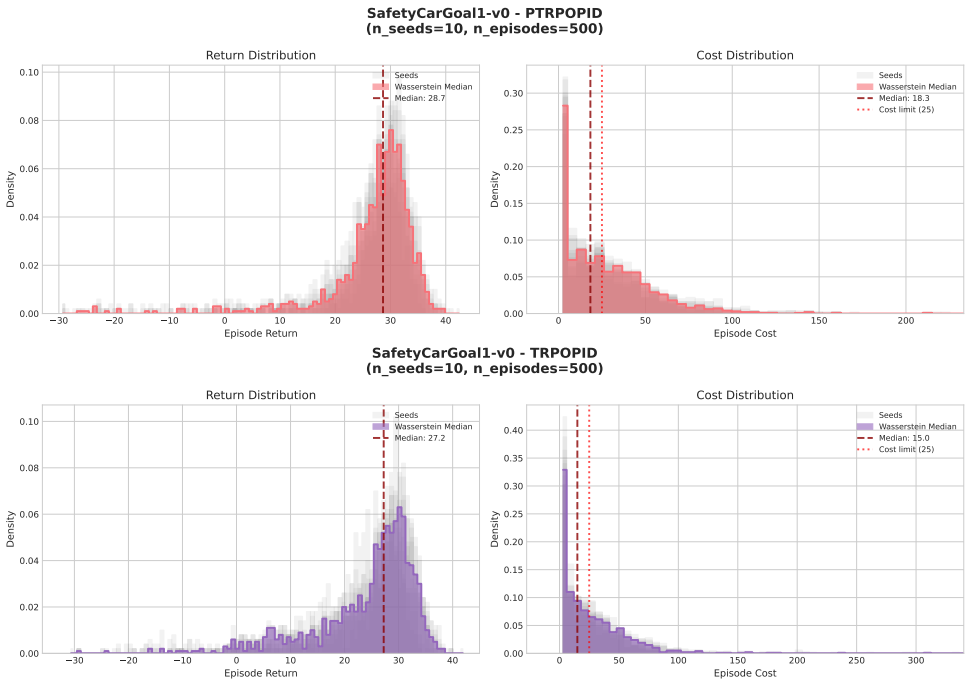


Figure 14: Deployment rollout distributions on `SafetyCarGoal1-v0` after 10^6 training steps. The phasic extension (PTRPOPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (TRPOPID) on the bottom.

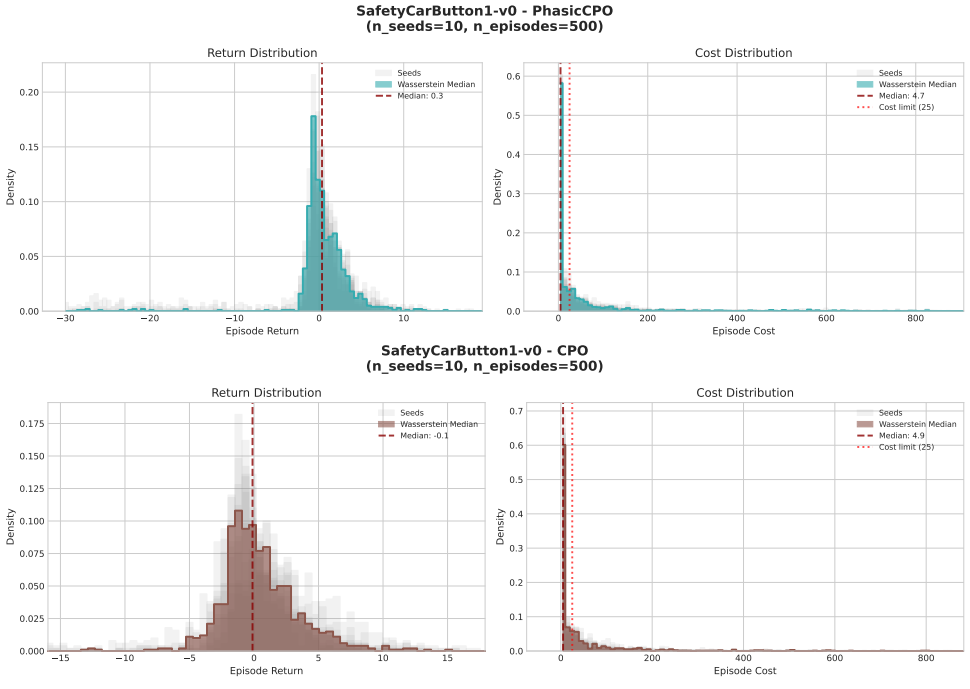


Figure 15: Deployment rollout distributions on `SafetyCarButton1-v0` after 10^6 training steps. The phasic extension (PhasicCPO) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPO) on the bottom.

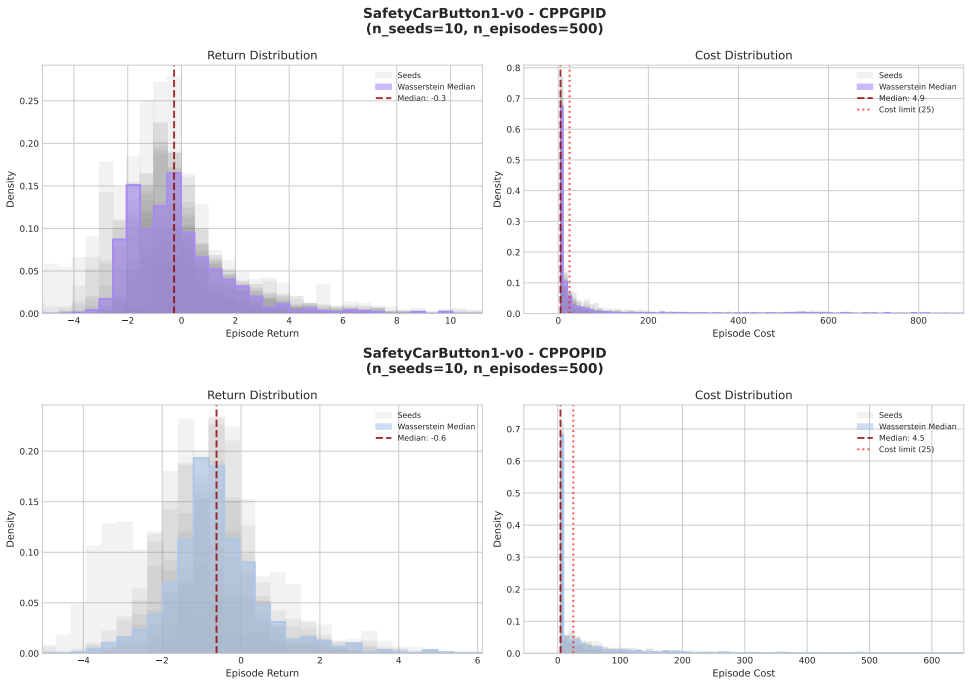


Figure 16: Deployment rollout distributions on `SafetyCarButton1-v0` after 10^6 training steps. The phasic extension (CPPGPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPPOPID) on the bottom.

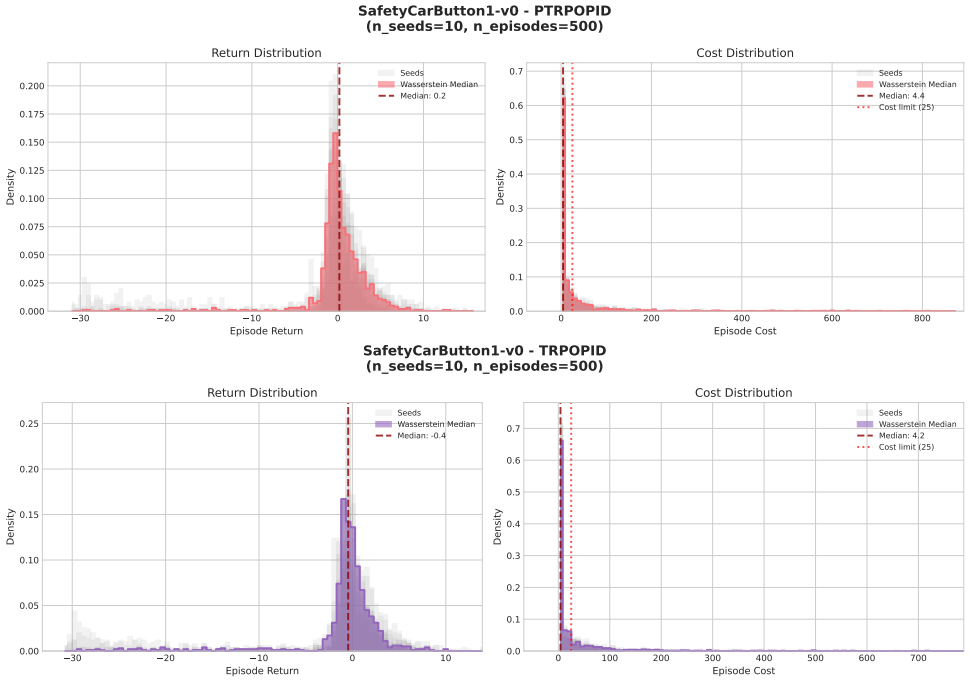


Figure 17: Deployment rollout distributions on `SafetyCarButton1-v0` after 10^6 training steps. The phasic extension (PTRPOPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (TRPOPID) on the bottom.

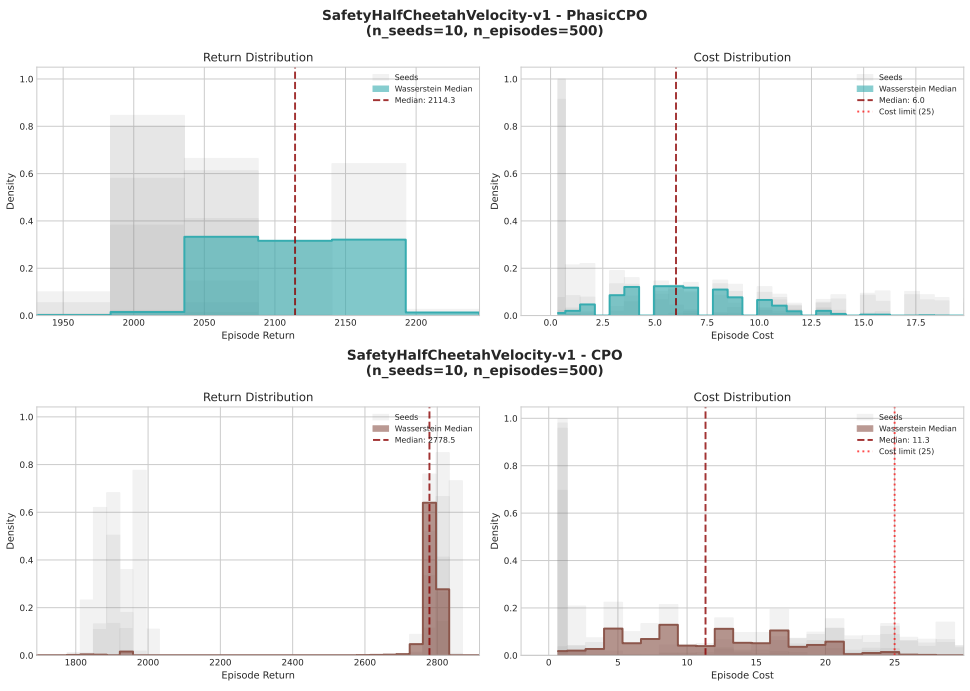


Figure 18: Deployment rollout distributions on `SafetyHalfCheetahVelocity-v1` after 10^6 training steps. The phasic extension (PhasicCPO) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPO) on the bottom.

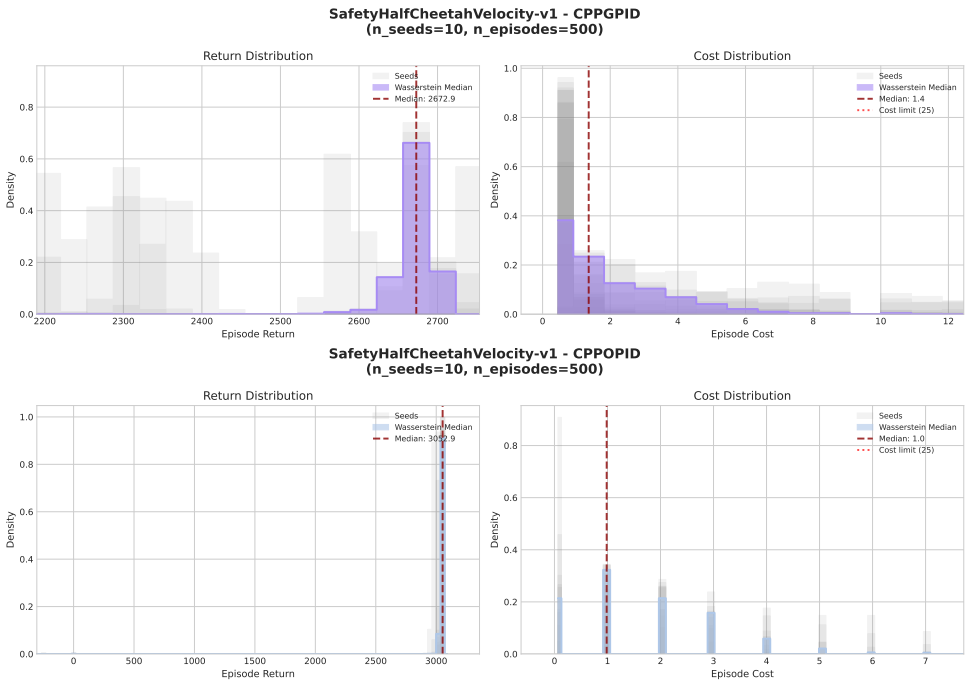


Figure 19: Deployment rollout distributions on `SafetyHalfCheetahVelocity-v1` after 10^6 training steps. The phasic extension (CPPGPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (CPPOPID) on the bottom.

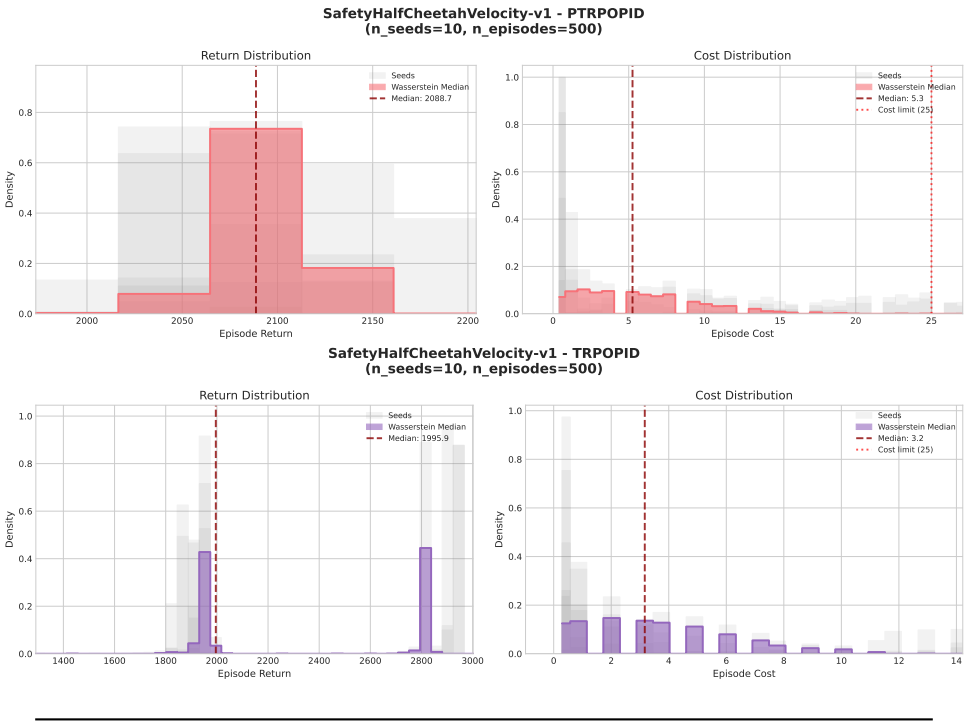


Figure 20: Deployment rollout distributions on `SafetyHalfCheetahVelocity-v1` after 10^6 training steps. The phasic extension (PTRPOPID) is displayed on top, visually bracketed with its vanilla baseline counterpart (TRPOPID) on the bottom.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing, helping with the implementation of methods, summarizing or analyzing existing literature.

We have not used generative AI tools to propose or refine hypotheses, design or provide feedback on research the methodology or experiments, assist with translation and clean and reformat dataset, support qualitative and thematic data analysis or to interpret results.

The generation of synthetic data sets, the development of theoretical models or conceptual frameworks, formulation of mathematical claims, providing of critical ingredients for proving mathematical claims, assisting in the writing of proofs are not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures or images and to identify relevant literature. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.