SafeExplorer: An Unbiased Policy Gradient for Reinforcement Learning with Recovery Interventions

Anonymous authors
Paper under double-blind review

Abstract

Training reinforcement-learning agents directly on physical robots makes every fall costly, since a fall can damage the platform and cannot be undone like a simulator reset; the goal is therefore to minimize falls during training rather than trade them off against return, as constrained Markov decision process (MDP) formulations do. A standard mitigation hands control to a separate *recovery policy* whenever the agent leaves a designer-specified *safe region* (a subset of state space it should stay within), but the resulting mixed-policy rollouts silently bias every on-policy update, and the importance-sampling correction that would remove this bias is ill-defined whenever the recovery policy is deterministic. We address this bias with a drop-in modification of proximal policy optimization (PPO). Its core is an unbiased policy-gradient estimator that uses the score function only at safe timesteps and never evaluates the recovery policy's density, so it stays valid even when the recovery policy is deterministic, exactly where importance sampling breaks, and it empirically dominates importance sampling even when the recovery policy is stochastic. Because the recovery policy still makes credit assignment slow near the safe-region boundary, two further components accelerate learning: a closed-form value for recovery-triggering states when dynamics and recovery are deterministic, and an imitation loss that copies recovery actions only when recovery *succeeds*. On a three-environment, five-seed benchmark, the resulting algorithm reduces training-time falls by factors of 233×, 48×, and 26× on HalfCheetah, Ant, and Unitree Go1 over standard PPO, while matching or exceeding PPO's final reward, and on Ant, where the recovery policy is unreliable, it is the only method that reaches 80% of the best final reward.

1 Introduction

Reinforcement learning (RL) policies often perform best when trained directly on the task they will be deployed on, but moving that training onto a physical robot is hard for a reason unrelated to asymptotic performance: learning requires failures, and on real hardware every failure has a price. A *fall*, a loss of balance that ends the episode, can damage the platform and cannot be undone the way a simulator reset can. Recent methods learn capable controllers by collecting hundreds of thousands of episodes in massively parallel simulation (Lee et al., 2020; Rudin et al., 2021; Agarwal et al., 2022), where these failures are free, but on a real robot they are not (Ibarz et al., 2021; Smith et al., 2023b). What gates real-world training is therefore not the final reward but the number of falls spent reaching it, and the goal of SafeExplorer is to keep that number small.

The common way to keep learning on real hardware is to pair the agent with a separate *recovery policy*, a controller that takes over whenever the agent leaves a designer-specified *safe region* and steers the system back toward a nominal state. Because the safe region is set conservatively, the agent leaves it routinely, so the recovery policy fires far more often than a fall would occur, converting most would-be falls into cheap, reversible interventions. The controller is easy to assemble from off-the-shelf parts, an model predictive control (MPC) solver (Pua & Khadiv, 2024), a soft actor-critic (SAC) expert (Haarnoja et al., 2018), or a hand-coded routine (Lee et al., 2019). But it carries a hidden cost: the data the agent collects is no longer

produced by the agent alone, but by a *mixed policy*, the main policy inside the safe region and the recovery outside it, so each rollout is a blend of two controllers rather than a sample from the one being improved.

This blend breaks the on-policy assumption. PPO (Schulman et al., 2017) and related methods estimate the gradient of the policy that produced the rollout, so they are valid only when the rollout distribution matches the policy being updated. The moment the recovery policy intervenes, every update is biased toward the recovery policy's behavior. Most safe-RL work leaves this mismatch unaddressed, updating as if all data came from the main policy (Srinivasan et al., 2020; Thananjeyan et al., 2021; Yang et al., 2022), or sidesteps it by shaping the reward to discourage entering recovery (Tessler et al., 2018; Stooke et al., 2020). The standard remedy, an importance-sampling correction (Degris et al., 2012), does not solve the problem either: a deterministic recovery (an MPC solver or a greedy SAC actor) produces a single action rather than a distribution, so the importance ratio has no density in its denominator and is undefined precisely where it is needed, the same obstruction that motivates deterministic policy gradients (Silver et al., 2014). The mechanism that makes real-world training feasible is thus the one that corrupts the learning signal, and a correct gradient must factor the recovery policy out rather than reweight through it.

Contributions. We close this gap with SafeExplorer, a practical algorithm built on a theoretical foundation for safe RL with a recovery policy. We make four contributions. *First*, we prove an unbiased policy-gradient theorem (Theorem 1) for any mixed policy that hands control to an external recovery policy on a subset of states. The gradient uses the main-policy score function only where the main policy acts and never evaluates the recovery policy's action density, so it applies unchanged to deterministic and stochastic recoveries, and empirically matches or outperforms importance sampling (IS) corrections in every measured regime, with orders-of-magnitude lower gradient variance where importance ratios explode (Remark 1). The theorem is general: safe-region intervention is the instance we develop (Corollary 2), and Jump-Start RL (Uchendu et al., 2023) (Corollary 8) and state-triggered shielded RL (Corollary 9) follow as corollaries of the same factorization. *Second*, we bound the gap between the mixed-policy return we optimize and the main-policy return we deploy (Theorem 4); this gap shrinks as the safe region grows and vanishes in the idealized limit where the region covers all reachable states. *Third*, when dynamics and recovery are deterministic, the value of a recovery-triggering state admits a closed-form expression (Proposition 6); used as the critic's target there, it provides dense, accurate supervision at the safe-region boundary where on-policy signal is otherwise scarce. *Fourth*, we add an outcome-gated compatibility regularizer (Definition 7) that pulls the main policy toward actions from *successful* recovery segments only, a warm start from behavior the recovery policy has shown to work; without the gate it reduces to a Dataset Aggregation (DAgger)-style behavioral-cloning loss (Ross et al., 2011) (Proposition 10).

Together these four pieces turn the recovery policy from a source of bias into a source of signal. On a three-environment, five-seed benchmark, they reduce training-time falls by 233×, 48×, and 26× on HalfCheetah, Ant, and Go1 over standard PPO, while matching or exceeding PPO's final reward. On Ant, where the recovery policy is unreliable, SafeExplorer is the only method that reaches the success threshold of 80% of the best reward.

2 Related Work

The setup we study, an agent paired with an external recovery policy that takes over outside a safe region, makes training rollouts follow a mixed policy that biases every on-policy update. We evaluate on legged locomotion, where end-to-end RL has advanced rapidly (Ha et al., 2025; Lee et al., 2020; Peng et al., 2018; Tan et al., 2018; Peng et al., 2020; Rudin et al., 2021; Agarwal et al., 2022; Kumar et al., 2021; Haarnoja et al., 2019; Bogdanovic et al., 2022) and real-world fine-tuning is now routine (Smith et al., 2023b; 2024; 2023a; Liu et al., 2024). These methods secure safety through reward design and sim-to-real robustness, but none confronts the bias a recovery policy injects into the on-policy gradient. Because that bias comes from the mixed rollout rather than any property of legged robots, our correction plugs into a locomotion pipeline without being tied to locomotion or any one robot.

Off-policy RL Correcting that bias looks like a standard off-policy problem, but off-policy policy-gradient and evaluation methods (Degris et al., 2012; Gu et al., 2017; Jiang & Li, 2016) rely on the importance ratio that Section 1 showed a deterministic recovery leaves undefined. Truncated-IS schemes such as V-trace

(Espeholt et al., 2018) and Retrace (Munos et al., 2016) cannot help, because the singularity sits upstream of the truncation. Our Theorem 1 factors the recovery measure out instead of reweighting, so it applies uniformly to deterministic and stochastic recoveries, MPC controllers (Chiu et al., 2022) included.

Mixed-policy and intervention data If reweighting is not the route, the alternative is to ask how prior work uses the same mixed-policy data, and rollouts in which a second policy intervenes are in fact common: they appear in DAgger (Ross et al., 2011; Kelly et al., 2019), in human-in-the-loop RL (Spencer et al., 2020; Saunders et al., 2018), and in Jump-Start RL (JSRL) (Uchendu et al., 2023). These either imitate the intervening policy or discard its transitions, so none extract a bias-corrected on-policy gradient from that data. Our masked gradient (Theorem 1) recovers that gradient directly, and our compatibility regularizer (Section 4.4) reduces in the JSRL setting to a state-dependent imitation loss with an outcome-based gate (Proposition 10).

Constraints and safety filters Delegating safety to an external recovery policy is itself the minority choice: the dominant lines of safe RL (García & Fernández, 2015) build safety into the policy rather than handing it to a separate controller. Constrained MDPs (CMDPs) encode safety as constraints on long-run cost (Altman, 1999), optimized via primal-dual schemes such as constrained policy optimization (CPO) (Achiam et al., 2017), reward-constrained policy optimization (Tessler et al., 2018), and PID-Lagrangian variants (Stooke et al., 2020). Because they trade cost against return, they suit soft-constraint settings where some failures are tolerable, whereas we treat falls as something to minimize rather than to budget against return. Closer to a hard guarantee, control barrier function (CBF) approaches (Ames et al., 2019) and shields (Junges et al., 2015; Alshiekh et al., 2018; Dalal et al., 2018; Srinivasan et al., 2020; Hasanbeig et al., 2020; Kang et al., 2022) project unsafe actions at execution time; this is exactly the case of our mechanism in which the intervention is a single-step projection, so our setup subsumes it as one instance. Closely related are the state-wise constrained MDP family (Zhao et al., 2023) and almost-surely safe RL (Sootla et al., 2022). Section 3 places all four formalisms in the constraint-formulation taxonomy of Wachi et al. (2024).

RL with a recovery policy The work closest to ours commits fully to that delegation: it uses an explicit recovery policy as an alternative to CMDP costs and pointwise filters, facing the same mixed-policy bias extended from the single-step projections of our shielded-RL corollary to multi-step recovery segments. Thananjeyan et al. (2021) train a safety critic and switch to the recovery policy when constraint risk crosses a threshold; they deliberately relabel recovery transitions with the task policy's proposed action, coherent for their off-policy Q-learning objective (Appendix B.7) but the source of the bias we correct once carried into an on-policy update. Yang et al. (2022) apply a similar template to legged locomotion with reward shaping, and Lee et al. (2019) train a model-free quadruped recovery controller, an example of the external recovery policy our masked gradient handles directly. Closest, Wagener et al. (2021) intervene on an advantage criterion and prove the intervened process safe, but still optimize the policy on the mixed rollouts without correcting the bias we address. Reverse-curriculum methods (Florensa et al., 2017) grow the region of starting states, and teacher-scheduled interventions (Turchetta et al., 2020) induce a safety curriculum, whereas safe-region curricula like ours grow the region the agent is permitted to enter. None of these treat the resulting distributional bias on the on-policy gradient, which our masked-gradient theorem (Theorem 1) addresses without requiring the recovery policy to have a density.

3 Problem Setting and Notation

We study RL in the *hard-safety regime* of the constraint-formulation taxonomy of Wachi et al. (2024). On a real robot a fall can break hardware, damage property, or injure a bystander, so the goal is to keep falls as rare as possible rather than to budget them against return.

The strictest formalism related to this regime is the state-wise constrained MDP family (Zhao et al., 2023), which demands constraint satisfaction at every step. We relax that requirement to *violation-minimization*, the model-free goal the survey identifies: without prior knowledge of the dynamics, hard state-wise safety cannot be guaranteed during training, so the target is to incur as few constraint violations (falls) as possible.

Among the ways to enforce safety in this regime (Section 2), ours is an external *multi-step* recovery policy that takes over whenever the state leaves a designer-specified safe region and returns the agent toward a

nominal configuration, unlike single-step action filters such as control-barrier projections and shields. We call this the *safe-region intervention* mechanism.

We instantiate the mechanism with a discounted MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma, T)$ with continuous state space $\mathcal{S}$ and action space $\mathcal{A}$, transition kernel P, reward $r \in [-r_{\max}, r_{\max}]$, discount $\gamma \in (0, 1)$, and episode horizon T. The analysis uses the infinite-horizon discounted convention (absorption at failure states); experiments truncate at $T = 1000$ steps. Let $\pi_\theta(a \mid s)$ be the main policy, stochastic and θ-parameterized. The recovery policy $\pi^{\text{rec}}(a \mid s)$ is an arbitrary state-conditioned measure that may be deterministic (MPC, greedy SAC) or stochastic, with no θ-dependence (its own fixed parameters, SAC weights or an MPC cost, are never optimized, so we suppress them). The safe region is the sublevel set $\mathcal{R}(d) := \{s : \mathcal{D}(s) \leq d\}$ of a task-space distance $\mathcal{D}$ to a nominal configuration, with $d \in [0, d_{\max}]$ grown over training by a curriculum schedule $d_{\text{iter}}(\cdot)$ ($d_{\max}$ a per-environment hyperparameter, Section 5). The *mixed policy* π_θ^{mix} follows π_θ inside $\mathcal{R}(d)$ and π^{rec} outside it. We give its general form and an unbiased gradient in Section 4.1. The trajectory distributions under π_θ alone and under π_θ^{mix} are $p_\theta(\tau) = p(s_1) \prod_t \pi_\theta(a_t \mid s_t) P(s_{t+1} \mid s_t, a_t)$ and $p_\theta^{\text{mix}}(\tau) = p(s_1) \prod_t \pi_\theta^{\text{mix}}(a_t \mid s_t) P(s_{t+1} \mid s_t, a_t)$. The *main-policy return* is $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[r(\tau)]$ with $r(\tau) := \sum_t \gamma^t r_t$. It is not directly observable during training, since rollouts are drawn from $p_\theta^{\text{mix}}(\tau)$, not $p_\theta(\tau)$. The *mixed-policy return*, the quantity training rollouts actually realize, is $J^{\text{mix}}(\theta) = \mathbb{E}_{\tau \sim p_\theta^{\text{mix}}(\tau)}[r(\tau)]$. Its state-conditional form, the *mixed-policy value* $V^{\pi_\theta^{\text{mix}}}(s) := \mathbb{E}_{\tau \sim p_\theta^{\text{mix}}(\tau)}[r(\tau) \mid s_1 = s]$, is what PPO's learned critic V_θ estimates. Discounted state-visitations under π_θ and π_θ^{mix} are ν^{π_θ} and $\nu^{\pi_\theta^{\text{mix}}}$, respectively. We use two rates, distinguished by their sampling distribution. The *recovery rate* $\alpha(\theta, d) := \Pr_{s \sim \nu^{\pi_\theta^{\text{mix}}}}[s \notin \mathcal{R}(d)]$ is sampled under the *mixed* distribution that training rollouts produce, hence *observable* as the fraction of rollout steps on which the recovery policy fired. The *out-of-region rate* $\beta(\theta, d) := \Pr_{s \sim \nu^{\pi_\theta}}[s \notin \mathcal{R}(d)]$ is sampled under the *main-only* distribution never deployed during training, hence *not directly observable*, yet it is the quantity that controls the gap in Theorem 4. A consolidated notation table is in Appendix B.2.

Together, these objects give the setup four operational features: binary set-membership safety, a possibly-deterministic recovery policy, multi-step recovery (a genuine mixed-policy distribution), and the unmodified main-policy objective, each addressed by one method component in Section 4 (Appendix B.3).

4 Method

SafeExplorer builds on PPO with four pieces, each removing a failure mode of recovery-using RL: (i) a *masked policy gradient* (Section 4.1) that removes the bias from treating recovery transitions as main-policy transitions; (ii) an *objective-gap bound* (Section 4.2) relating the training (mixed-policy) return to the deployment (main-policy) return and showing a safe-region curriculum closes the gap; (iii) an *analytic recovery value* (Section 4.3) that replaces the learned critic at recovery-triggering states with a closed form under deterministic dynamics; and (iv) an *outcome-gated compatibility regularizer* (Section 4.4) that imitates recovery only after successful segments. We develop each in turn.

4.1 Unbiased policy gradient via partition masking

To make the most of training data, we want every rollout to contribute to the task gradient, even those where the recovery policy took over for long stretches. The obstacle is that treating recovery transitions as if the main policy generated them biases the on-policy update, and the IS fix breaks down whenever the recovery policy is deterministic. π_θ, a Gaussian in our PPO implementation, has a differentiable log-density, whereas a deterministic π^{rec} is a Dirac measure with no density. Because this obstacle arises whenever an external, θ-independent policy takes over on a subset of states, not only in safety, we solve it in full generality first and then specialize to safe-region intervention.

Setup. We generalize the mixed policy of Section 3. Let $\mathcal{M} \subseteq \mathcal{S}$ be a designer-specified *main-policy set* (where π_θ acts), and let $\mu(\cdot \mid s)$ be any state-conditioned action measure that is *independent of* θ (it may be deterministic or stochastic). The resulting *partition policy* is

$$\pi_\theta^{\text{mix}}(a \mid s) = \begin{cases} \pi_\theta(a \mid s) & s \in \mathcal{M}, \\ \mu(a \mid s) & s \notin \mathcal{M}, \end{cases} \tag{1}$$

and the trajectory return $J^{\text{mix}}(\theta) = \mathbb{E}_{\tau\sim p_\theta^{\text{mix}}}[r(\tau)]$.

Assumption 1 (Regularity)**.** **(R1)** For every (s, a) on the support of p_θ^{mix} with $s \in \mathcal{M}$, $\pi_\theta(a \mid s) > 0$ and $\theta \mapsto \log \pi_\theta(a \mid s)$ is differentiable. **(R2)** $T < \infty$, or $|r_t| \le r_{\max}$ uniformly. **(R3)** Differentiation and integration commute in Equation (2).

For diagonal-Gaussian π_θ with bounded mean and log-std, and bounded reward, (R1) to (R3) all hold.

Theorem 1 (Unbiased partition-policy gradient)**.** *Under Assumption 1,*

$$\nabla_\theta J^{\text{mix}}(\theta) = \mathbb{E}_{\tau\sim p_\theta^{\text{mix}}}\left[\left(\sum_{t=1}^{T} \mathbb{1}[s_t \in \mathcal{M}]\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right) r(\tau)\right]. \tag{2}$$

Here $\mathbb{1}[\cdot]$ is the indicator function, so the sum runs only over timesteps with $s_t \in \mathcal{M}$.

We prove this in Appendix A.1: because μ is θ-independent, its factor in the trajectory density vanishes under differentiation, so the masked gradient never evaluates μ's density and holds for any μ. This is exactly the case the intra-option policy-gradient results (Sutton et al., 1999; Bacon et al., 2017) assume away, extending the factorization to the density-free external controllers they exclude, and it is why we never need the IS correction.

Remark 1 (Masking dominates importance sampling)*.* For stochastic μ the per-step IS estimator decomposes into the masked estimator plus a term carried by the weights π_θ/μ at recovery steps; that term has a generically nonzero mean under a genuine advantage, so per-step IS is in general biased for $\nabla_\theta J^{\text{mix}}$, whereas the masked estimator is exactly unbiased (Theorem 1; decomposition in Equation (15), Appendix E.13). The weights also degenerate, since recovery acts unlike the main policy by design: the measured IS gradient variance exceeds the masked variance by up to 13 orders of magnitude (an empirical measurement, not a theorem), and truncated IS loses 40% reward on Ant, while masking is identical or better in every regime we test. Masking, not IS, is the estimator SafeExplorer optimizes (Corollary 3).

Specializations. Different choices of $(\mathcal{M}, \mu)$ specialize Theorem 1 to concrete settings.

Corollary 2 (Safe-region intervention)**.** *Take $\mathcal{M} = \mathcal{R}$ (the safe region) and $\mu = \pi^{\text{rec}}$ (the recovery policy, θ-independent by setup), recovering the mixed policy π_θ^{mix} of Section 3 with return $J^{\text{mix}}(\theta)$. Then Theorem 1 gives $\nabla_\theta J^{\text{mix}}(\theta) = \mathbb{E}_{\tau\sim p_\theta^{\text{mix}}(\tau)}[(\sum_t \mathbb{1}[s_t \in \mathcal{R}]\, \nabla_\theta \log \pi_\theta(a_t \mid s_t))\, r(\tau)]$.*

The same factorization yields Jump-Start RL (Uchendu et al., 2023) and state-triggered shielded RL as corollaries (Corollaries 8 and 9, Appendix A.1). The rest of the paper develops safe-region intervention (Corollary 2).

Corollary 3 (PPO surrogate with safe-step masking)**.** *Specializing Corollary 2 to the clipped PPO surrogate, the empirical masked-gradient estimator is*

$$\widehat{\nabla J^{\text{mix}}}(\theta) = \frac{1}{|T_{\mathcal{R}}|} \sum_{t:s_t\in\mathcal{R}} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}_t^{\text{full}}, \tag{3}$$

where $\hat{A}_t^{\text{full}}$ is the generalized advantage estimation (GAE) (Schulman et al., 2016) advantage over the full mixed-policy trajectory (recovery segments enter through the return exactly as in Theorem 1, while the mask restricts only the score function) and $|T_{\mathcal{R}}| = |\{t : s_t \in \mathcal{R}\}|$. The clipped PPO surrogate is

$$L_{\text{safe}}^{\text{PPO}}(\theta) = -\frac{1}{|T_{\mathcal{R}}|} \sum_{t:s_t\in\mathcal{R}} \min\Big(\rho_t(\theta)\, \hat{A}_t^{\text{full}},\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\, \hat{A}_t^{\text{full}}\Big), \tag{4}$$

with $\rho_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$, well-defined since both are the main policy.

The masked score-function term is exactly unbiased for $\nabla_\theta J^{\text{mix}}$ (Theorem 1). Relative to the standard clipped-PPO surrogate over the same rollouts, Equation (3) changes only two things: the sums run over safe steps, and the normalizer is $|T_{\mathcal{R}}|$, the number of such steps in a rollout and hence random, which the curriculum on d keeps large in practice (Section 5, Appendix B.5). GAE, the clipping, and thus PPO's stability heuristics are otherwise unchanged.

4.2 Bounding the gap between mixed-policy and main-policy returns

Theorem 1's gradient targets the *mixed-policy return* $J^{\mathrm{mix}}(\theta)$, the quantity training rollouts realize, but the deployment-relevant quantity is the *main-policy return* $J(\theta)$, what π_θ earns alone, which training never samples directly. Optimizing one while caring about the other is only safe if the two cannot drift far apart, so this section bounds $|J^{\mathrm{mix}} - J|$ and shows it is exactly zero under reading (R-A) below.

The two returns differ only on the steps where π_θ acts but $\pi_\theta^{\mathrm{mix}}$ would have called the recovery policy, that is, the steps where π_θ alone leaves the safe region. The gap is therefore controlled by how reliably π_θ keeps itself inside $\mathcal{R}$, which we make precise with a one-step invariance property.

Assumption 2 (Approximate π_θ-invariance of $\mathcal{R}$)**.** There exists $\eta \geq 0$ such that, starting from any reachable $s \in \mathcal{R}$, π_θ stays in $\mathcal{R}$ at the next step with probability at least $1 - \eta$ uniformly over time.

This one-step set-invariance property holds with $\eta = 0$ exactly in the idealized limit (R-A) below and approximately in our setting (R-B).

Summed over a discounted trajectory, this one-step slack η controls the out-of-region rate $\beta(\theta, d)$ (Section 3), which inherits a bound $\beta(\theta, d) \leq \eta\gamma/(1-\gamma) \leq \eta/(1-\gamma)$ whenever episodes start inside the region, as ours do (Appendix A.2).

Theorem 4 (Objective-gap bound)**.** *Under Assumption 1,*

$$|J(\theta) - J^{\mathrm{mix}}(\theta)| \leq \frac{2\, r_{\max}}{(1-\gamma)^2}\, \beta(\theta, d). \tag{5}$$

The proof (Appendix A.2) applies the Performance Difference Lemma (Kakade & Langford, 2002). The advantage of π_θ against $\pi_\theta^{\mathrm{mix}}$ vanishes inside $\mathcal{R}$, where the two policies agree, and is bounded by $2r_{\max}/(1-\gamma)$ outside it, so the gap scales with the main-policy out-of-region rate $\beta(\theta, d)$. The bound is non-vacuous only for $\beta(\theta, d) < 1 - \gamma$ ($\eta_\star < (1-\gamma)^2$ in Corollary 5); we read it structurally (Appendix E.12).

How far Equation (5) can be pushed as the curriculum radius d_{iter} grows to $d_{\max}$ depends on the largest safe region it reaches, $\mathcal{R}(d_{\max})$, which admits two readings.

Corollary 5 (Conditional fixed-point)**.** ***(Exact, R-A.)*** *If $\mathcal{R}(d_{\max})$ covers the reachable state space, then at $d_{\mathrm{iter}} = d_{\max}$, $\pi_\theta \equiv \pi_\theta^{\mathrm{mix}}$ on the reachable support of $p_\theta(\tau)$, $\beta(\theta, d_{\max}) = 0$, and $J^{\mathrm{mix}}(\theta) = J(\theta)$ exactly.* ***(Approximate, R-B.)*** *Otherwise $|J(\theta) - J^{\mathrm{mix}}(\theta)| \leq 2r_{\max}\eta_\star/(1-\gamma)^3$, where $\eta_\star = \eta(\theta, d_{\max}) \geq 0$ is the env-dependent invariance slack that (R-B) does not force to zero.*

The proof specializes Theorem 4 to the two readings of $\mathcal{R}(d_{\max})$ (Appendix A.3). Our setting is (R-B): $\mathcal{R}(d_{\max})$ is a tuned hyperparameter strictly inside the reachable set, so the bound is approximate, the theory no longer forcing $\eta = 0$, though the curriculum can still drive it low in practice. Theorem 4 then says any reduction in $\beta(\theta, d)$ tightens the gap proportionally, which the linear schedule $d_{\mathrm{iter}} : d_0 \to d_{\max}$ achieves, with the observable $\alpha(\theta, d)$ as its diagnostic (Appendix E.12). Falls are orthogonal to d, and those still possible under (R-B) are what the compatibility regularizer (Section 4.4) targets. Appendix A.7 sketches why $\beta(\theta, d)$ is hard to bound from *observable* quantities in continuous-action settings.

4.3 Analytic recovery value

The masked gradient fixes the policy gradient, but it leaves a second error untouched: the critic. A learned critic trained over the recovery segments bootstraps its value targets *through* them, so even with a corrected gradient the value-target error persists; empirically, the learned-V variant under-performs every other variant on HalfCheetah. Replacing the learned critic at recovery-triggering states with the analytic value derived below removes this error, and is the largest single-ingredient reward gain on HalfCheetah at no cost in falls (on Ant and Go1 the outcome gate contributes more; quantified in Section 8).

When dynamics and recovery are both deterministic, the practically common case (MPC, greedy SAC), the value at a recovery-triggering state has a closed form: the recovery segment unrolls along a single fixed path, so its contribution is the return along that path rather than something the critic must learn.

Proposition 6 (Analytic recovery value)**.** *Suppose dynamics P and recovery π^{rec} are both deterministic. Let s_t be a recovery-triggering state, $k \geq 1$ the number of recovery steps until either re-entry into $\mathcal{R}$ at $t+k$ (*success*) or termination at $t+k$ (*failure*), and $G_{t,k} = \sum_{j=0}^{k-1} \gamma^j r_{t+j}$ the realized segment return. Then*

$$V^{\pi_\theta^{\text{mix}}}(s_t) = \begin{cases} G_{t,k} + \gamma^k V^{\pi_\theta^{\text{mix}}}(s_{t+k}) & \textit{success}, \\ G_{t,k} & \textit{failure}. \end{cases} \tag{6}$$

Under our convention that recovery accrues zero per-step task reward and failure incurs a one-time terminal failure reward r_{term}, the segment return $G_{t,k}$ is 0 on success and $\gamma^{k-1} r_{\text{term}}$ on failure. In practice we bootstrap the re-entry value with the learned critic V_θ, writing $\gamma^k V_\theta(s_{t+k})$ into the buffer on success (Algorithm 1). This carries no Monte-Carlo variance under deterministic P and π^{rec}.

The proof is direct (Appendix A.4): under deterministic dynamics and recovery, the post-trigger trajectory is a single deterministic path of length k, so its return is the realized return.[1]

In practice we write Equation (6) into the rollout buffer before GAE, which then proceeds normally and feeds the masked gradient (Corollary 3) only at safe steps (Appendices B and B.5).

4.4 The compatibility regularizer

The masked gradient and analytic value together define a provably unbiased on-policy update for $J^{\text{mix}}(\theta)$, but unbiasedness alone does not make the agent self-sufficient: neither component pulls π_θ toward behaving like recovery at unsafe states. This matters precisely when recovery is unreliable (Section 7.2): an unbiased update over repeated failed recoveries teaches the policy nothing about avoiding those states unaided. We close this gap with a lightweight imitation-from-recovery loss with an outcome-conditioned weight. As an imitation signal rather than an importance correction, it needs no recovery density and covers deterministic controllers. The closest methods, DAgger (Ross et al., 2011) and JSRL (Uchendu et al., 2023), do not gate the imitation signal by segment outcome, and self-imitation learning (Oh et al., 2018), which does, imitates the agent's own past actions rather than an external recovery policy, the distinction we now make precise.

Definition 7 (Compatibility regularizer)**.** Let $a_t^{\text{rec}} = \pi^{\text{rec}}(s_t)$ be the recovery action executed at unsafe step t. The trajectory-level compatibility score is

$$C_\theta(\tau) = \prod_{t: s_t \notin \mathcal{R}} \pi_\theta(a_t^{\text{rec}} \mid s_t), \quad \log C_\theta(\tau) = \sum_{t: s_t \notin \mathcal{R}} \log \pi_\theta(a_t^{\text{rec}} \mid s_t). \tag{7}$$

A high $C_\theta(\tau)$ measures exactly the agreement we want, π_θ assigning high likelihood to the recovery actions at the states where it intervened. Raising it indiscriminately is harmful: a failed recovery (segment ends in termination) is exactly what π_θ should *not* imitate. We therefore gate the imitation signal by the realized outcome of each segment. Let $\sigma_k \in \{0, 1\}$ indicate success of recovery segment k (re-entry into $\mathcal{R}$), and $\text{seg}(t)$ the segment containing step t. Reading success off the realized rollout keeps the gate applicable whether recovery is deterministic or stochastic. The *hard outcome gate* is

$$w_t^{\text{hard}} = \sigma_{\text{seg}(t)}, \quad t : s_t \notin \mathcal{R}. \tag{8}$$

Two boundary cases follow the same reading: a segment cut off by rollout truncation counts as a tentative success with a bootstrapped value, and an immediate re-exit opens a new segment. A signed soft variant based on per-step value change is ablated in Appendix B.4; the hard gate dominates empirically, so $w_t = w_t^{\text{hard}}$ throughout unless noted.

Putting the score and its gate together, the regularizer is the outcome-gated form of $-\log C_\theta(\tau)$ from Equation (7), added to the PPO loss with coefficient λ_{compat}:

$$L^{\text{compat}}(\theta) = -\frac{\lambda_{\text{compat}}}{N_{\text{rec}}} \sum_{t: s_t \notin \mathcal{R}} w_t \log \pi_\theta(a_t^{\text{rec}} \mid s_t), \quad L(\theta) = L_{\text{safe}}^{\text{PPO}} + c_v L^V + L^{\text{compat}}. \tag{9}$$

[1] Theorems 1 and 4 hold for any bounded reward, covering this training-signal convention; the induced train-to-deploy return gap is controlled by the same $\beta(\theta, d)$ (Appendix A). Reported returns always use the unmodified task reward (Section 6).

Here N_{rec} counts the recovery-controlled minibatch steps (Appendix B.2; per-segment normalization ablated in Appendix E.9), and L^V is the standard PPO critic loss with coefficient c_v (entropy bonus retained, $c_e = 0$ by default; Appendix B).

L^{compat} is a supervised auxiliary term, deliberately outside the unbiased J^{mix} gradient: it trades a tunable amount of bias, scaled by λ_{compat}, for a signal the unbiased components cannot supply, pulling π_θ toward the actions of *successful* recovery segments. That trade pays off exactly where it should: the term costs a little reward on HalfCheetah but adds substantially on Go1 and Ant (+1121 and +2683 reward over the gate-free variant, Section 8).

The gate is all that separates this term from standard imitation: always on ($\sigma_k \equiv 1$), L^{compat} reduces to DAgger-style (Ross et al., 2011) cloning on the recovery-controlled states, π^{rec} as teacher (Proposition 10, Appendix A.6); the gate echoes the Q-filter of Nair et al. (2018), reading a realized segment outcome instead of a learned value estimate.

5 Algorithm

SafeExplorer assembles the three algorithmic pieces developed in the previous sections, the masked policy gradient (Theorem 1, Corollary 3), the analytic recovery value (Proposition 6), and the hard outcome-gated compatibility regularizer (Section 4.4), into a single PPO update (Algorithm 1); the fourth piece, the objective-gap bound (Theorem 4), needs no implementation beyond the curriculum below. Because each piece intervenes only where the recovery policy acts, the resulting algorithm is a small edit to a standard PPO implementation: the changes touch only the rollout's recovery branch and the loss, everything else is unchanged. That locality keeps the method cheap: the added cost is one recovery-policy forward pass per triggered step plus segment bookkeeping (runtimes in Appendix C).

This leaves one moving part, the safe-region radius that decides when recovery is triggered, which we anneal over training with a linear curriculum (Bengio et al., 2009; Narvekar et al., 2020): $d_{\text{iter}}(u) = d_0 + \frac{u-1}{N} d_{\max}$, growing the region from a tight d_0 to the per-environment $d_{\max}$ over the N updates (d_0, $d_{\max}$, N in Appendix C). Appendix E.7 ablates logarithmic, step, and constant schedules on Ant, our least reliable recovery policy, where the linear schedule attains the highest reward. This annealing is the practical mechanism by which the recovery rate $\alpha(\theta, d)$ falls over training (Appendix E.12), an empirical diagnostic for the gap-tightening predicted by Theorem 4.

6 Experimental Setup

Our experiments evaluate three claims. The first is the *safety-reward trade-off* of SafeExplorer against standard PPO, recovery-using baselines, and CMDP baselines, judged on both falls and reward. The second is the *unreliable-recovery regime* that most motivates the method, where the recovery policy itself often fails and the trade-off is hardest to win. The third asks *which ingredient* drives the gains: the unbiased masked gradient (Theorem 1), the analytic recovery value (Proposition 6), or the outcome-gated regularizer (Definition 7). A single metric adjudicates all three: the number of training falls a method incurs before first reaching task success, where success is 80% of the best final reward attained by any method in that environment. On Ant and Go1 that best method is SafeExplorer, so the reward winner partly sets its own bar; Table 1 therefore reports the underlying rewards and fall counts separately.

Environments. We evaluate on three continuous-control locomotion environments: HalfCheetah and Ant (custom MuJoCo (Todorov et al., 2012)/Gymnasium (Towers et al., 2023) variants with a configurable safe-region indicator; full specification in Appendix D), and Go1, a Unitree Go1 quadruped built on the `mujoco_menagerie` model (Zakka et al., 2022) with a velocity-tracking reward adapted from `legged_gym` (Rudin et al., 2021). The mechanism is identical across all three: the distance $\mathcal{D}$ defining $\mathcal{R}(d)$ is a (z, tilt) task-space distance to the nominal upright pose (exact formulas in Appendix D.2), and each environment keeps its standard locomotion reward, zeroed only during recovery (Section 4.3). The environments differ in how hard they make safety: HalfCheetah (planar, strong recovery) sets the baseline, Ant (3D, weaker

recovery) tests whether the compatibility regularizer compensates for an unreliable fall-back, and Go1 (12-degrees of freedom (DoF) quadruped) scales the claim to a higher-dimensional, multi-contact platform.

Recovery policies. Each environment uses a separately pre-trained SAC (Haarnoja et al., 2018) actor as π^{rec}: a stand-and-stabilize objective on its own environment, task terms disabled, no demonstrations, aggressively randomized resets (recipes in Appendix D.3). At rollout time it takes the SAC mean action, deterministic, so it exercises the no-density case the masked gradient is built for (Theorem 1). The Ant recovery policy is meaningfully weaker than the others, the most informative axis of variation: its strong control-magnitude penalty caps how aggressively it can correct, and the harder 3D stabilization needs those corrections (Appendix D).

The remaining choice is the radius $d_{\max}$, the one quantity a skeptic might suspect was tuned. Unbiasedness holds for any $d_{\max}$ (Theorem 1), so no radius can manufacture statistical validity, though it still shapes results through the gap bound (Theorem 4) and the curriculum. We set each $d_{\max}$ (Appendix C) by a qualitative criterion, that the curriculum lowers the observable intervention rate (Appendix E.12), without a quantitative search. The fixed-d sweep (Table 10) shows sensitivity to a *constant* radius that the curriculum mostly recovers (Appendix E.7); no systematic sweep varies $d_{\max}$ under the curriculum, a limitation (Appendix E).

Variants compared. SafeExplorer (Algorithm 1) is compared against three external baselines spanning the established alternatives identified in Section 2. Standard PPO is the no-recovery extreme. Recovery RL (Thananjeyan et al., 2021) and Safe Legged (Yang et al., 2022) are *on-policy ports* of the two published recovery-using alternatives: both run inside the same PPO loop, sharing SafeExplorer's safe-region trigger and frozen SAC recovery policy, and each isolates its published method's data-handling rule at recovery steps, action relabeling for Recovery RL versus relabeling plus a fixed reward penalty of 1 for Safe Legged (exact rules and deviations in Appendix B.7). A controlled ablation isolating each ingredient of SafeExplorer is reported in Section 8; additional soft-gate variants are in Appendix E.4.

CMDP baselines. To position SafeExplorer against the standard no-recovery alternative, we compare to two CMDP solvers, CPO (Achiam et al., 2017) and PPO-Lagrangian (Ray et al., 2019; Stooke et al., 2020), which act with a single policy and encode safety as a constraint on an indicator cost (1 per fall). We run both via OmniSafe (Ji et al., 2024) at settings matched to SafeExplorer, except the solvers' default observation and reward normalization, which SafeExplorer does not use, favoring the baselines (Appendix E.14). Section 7.3 analyzes their constraint feasibility.

Training and evaluation. We train each variant for 8M environment steps on HalfCheetah and Ant and 16M on Go1 (its higher-dimensional action space needs a longer horizon), with five random seeds. Final reward is the per-episode mean of the last 10% of training, computed from the unmodified task reward, not the recovery-zeroed learning signal; falls are the cumulative count over the full training run, a fall being a per-environment unhealthy termination (Table 6), never a time-limit truncation. Tables report mean $\pm$ standard deviation over seeds; the headline and ablation plots aggregate with interquartile means and 95% stratified-bootstrap confidence intervals (Agarwal et al., 2021). A method's falls-to-success is the cumulative falls before its trailing-mean reward first crosses the success bar; a method that never crosses it is counted as not reaching success, and falls-to-success aggregates use the succeeding seeds only (the k/n labels in Figure 1). Shared PPO hyperparameters follow CleanRL's defaults (Huang et al., 2022) and are listed in Table 4.

7 Main Results

7.1 SafeExplorer reaches task success with the fewest falls

Addressing our first claim, we compare all methods on the three environments: Figure 1 reports falls-to-success with the per-method solve counts k/n, Table 1 the separate final-reward and total-training-fall figures (different quantities, since total falls span the whole run, not just the climb to success), and Figure 2 the per-metric rliable interquartile-mean view. SafeExplorer reaches success with the fewest training falls in every environment, an interquartile mean of 12 falls on HalfCheetah, 17 on Ant, and 118 on Go1. The advantage is most stark on Ant, where it is the only method to reach success at all, on all five seeds, while

every baseline fails to cross the bar on any seed (quantified in Section 7.2). The separation also widens with the difficulty of staying safe: on Go1 the next-cheapest method after SafeExplorer to reach success at all is Recovery RL on a single seed (309 falls), with PPO-Lagrangian and PPO at 9,658 and 15,320. The same lead shows up in the table's lifetime view: measured as total training falls (Table 1), SafeExplorer reduces falls over PPO by **233**× on HalfCheetah, **48**× on Ant, and **26**× on Go1 while matching or beating PPO's final reward.

Table 1: Final reward (mean over the last 10% of training, on the same unmodified task reward for every method) and total training-time falls for SafeExplorer and all baselines. For recovery-using methods reward is the training-time mixed-policy return, including recovery actions (recovery-disabled return for SafeExplorer: Appendix E.12). CPO and PPO-Lagrangian are CMDP baselines with a per-fall cost (Section 6). Bold marks the best value per column; on HalfCheetah falls, SafeExplorer, PPO-Lagrangian, and CPO are within seed noise. This is the canonical body-protocol batch anchoring the headline fall-reduction ratios; the appendix sensitivity tables use separate re-run batches (batch provenance in Appendix E).

	HalfCheetah		Ant		Go1	
Method	Reward ↑	Falls ↓	Reward ↑	Falls ↓	Reward ↑	Falls ↓
PPO (no recovery)	3246 ± 521	3311 ± 940	308 ± 165	36467 ± 14598	3347 ± 215	18683 ± 1403
Recovery RL	1951 ± 1423	947 ± 1831	76 ± 25	53535 ± 7611	3136 ± 511	1689 ± 1379
Safe Legged	$\mathbf{4054 \pm 604}$	1552 ± 1084	83 ± 20	53548 ± 11682	2347 ± 353	3022 ± 1225
CPO	2835 ± 403	23 ± 30	1047 ± 106	899 ± 117	359 ± 48	39613 ± 2498
PPO-Lagrangian	2939 ± 414	18 ± 19	1339 ± 402	5094 ± 2774	3915 ± 953	12572 ± 1636
SafeExplorer (ours)	3581 ± 971	$\mathbf{14 \pm 8}$	$\mathbf{2857 \pm 129}$	$\mathbf{754 \pm 463}$	$\mathbf{5594 \pm 177}$	$\mathbf{731 \pm 177}$

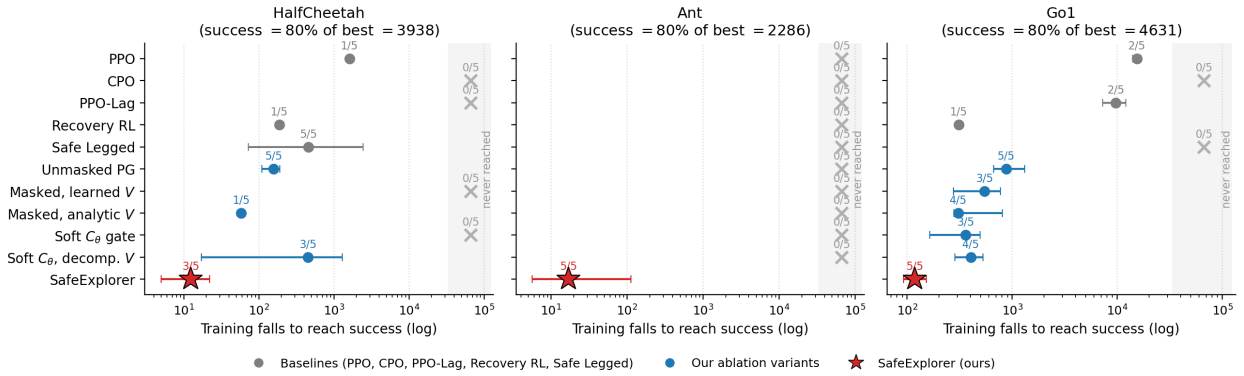

Figure 1: Training falls to reach task success (success = 80% of the best final reward in each environment), interquartile mean with 95% bootstrap confidence intervals (Agarwal et al., 2021); log axis, fewer falls is better. Labels show the seeds reaching success (k/n); never-reachers are marked at right. Baselines are gray, SafeExplorer ablation variants (Section 8) blue, and SafeExplorer the red star, which reaches success with the fewest falls in every environment and is the only method to reach it on Ant.

Reading the environments one at a time, the same ranking holds but the reason shifts with how hard safety is. On HalfCheetah, where the recovery policy is reliable, PPO is competitive on reward but pays for it in falls, while SafeExplorer matches that reward at a fraction of the falls. On Go1, the larger action space and longer horizon make PPO highly unsafe, while Recovery RL and Safe Legged control falls only at substantial reward cost and SafeExplorer attains both the highest reward and the lowest fall count. Ant is the informative exception: the ranking still favors SafeExplorer, but it behaves differently because the recovery policy itself is unreliable, the case we isolate next in Section 7.2.

This dependence on recovery is exactly what Theorem 4 predicts: Appendix E.12 confirms on body-matched 5-seed runs that the gap $J^{\text{mix}}(\theta_k) - J(\theta_k)$ tracks the recovery rate $\alpha(\theta, d)$, closing to seed noise on HalfChee-

TAH and ANT, where the policy becomes self-stable, and settling at a proportional residual on GO1, where it does not.

Per-environment learning curves are deferred to Figure 3 in Appendix E.

7.2 The unreliable-recovery regime: Ant

This subsection addresses our second claim, the unreliable-recovery regime: Section 7.1 cannot show *why* SafeExplorer remains effective where the recovery policy itself fails, so we isolate the one environment where that happens, ANT, whose SAC recovery is markedly less reliable. Replaying the recovery from the states a trained policy actually drives it into, it completes the segment only 70 to 91% of the time on ANT, against 97 to 100% on HALFCHEETAH and GO1 (one representative run per environment; Table 7). The recovery-based baselines reveal the latter: PPO, Recovery RL, and Safe Legged all fail catastrophically (36k to 54k falls, reward ≤ 308). The mechanism is a feedback cycle, because when the recovery policy is unreliable, any method that relies on it *reactively* keeps re-entering unsafe states, recovery keeps failing, and the agent never learns to avoid them unaided. SafeExplorer breaks that cycle and alone reaches the 2286 success bar, at 2857 reward and 754 falls, an order of magnitude fewer falls than any other recovery-using method; the strongest baselines, PPO-Lagrangian and CPO, reach 1339 and 1047. The ablations pinpoint why: every variant that drops the hard outcome gate, including the soft-gate variants, also fails on ANT (Section 8); the gate, not the masked gradient or the analytic value, supplies the fallback skill when recovery is unreliable.

7.3 Comparison to CMDP baselines

This subsection completes our first claim by comparing SafeExplorer against the CMDP alternative to a recovery policy, which casts safety as a constraint and solves the resulting CMDP rather than intervening on the trajectory. CPO and PPO-Lagrangian, the trust-region and Lagrangian solvers, appear in Table 1 and Figure 1 with a cost of 1 per fall, at settings identical to SafeExplorer. SafeExplorer attains the highest reward of the three in every environment and the fewest total training falls on ANT and GO1, with all three methods low and within seed noise on HALFCHEETAH (Table 1). The more revealing comparison is constraint feasibility: whether each solver keeps the falls it promised to.

The constrained solvers' poor safety on the hard environments is not under-tuning. Each CMDP solver is configured with a cost limit on falls, as its formulation requires, and meets it on HALFCHEETAH where staying safe is easy, but *violates its own limit* on ANT and GO1 by up to 19×: CPO converges to an episodic cost of 0.96 on GO1 against the 0.05 budget it was set, and even incurs more GO1 falls than unconstrained PPO at a tenth of the reward (Appendix E.14). Constraint satisfaction thus holds where safety is easy and fails exactly where the safe-region intervention is needed. The difference is one of mechanism: the CMDP baselines act alone and only penalize the violation after the fact, whereas SafeExplorer acts on the safe region directly, which is why it keeps falls low where the constrained solvers cannot.

8 Ablation Analysis

This section addresses our third claim, which ingredient drives SafeExplorer's gains over prior recovery-using methods. We isolate each of its three new ingredients with a ladder of controlled variants that share the recovery policy and curriculum and differ only at recovery steps. The ladder starts from the biased update those methods use, which keeps recovery transitions as if the main policy had generated them (a learned-critic on-policy gradient), and turns on one ingredient at a time: *Unmasked PG* is that biased baseline; *masked, learned V* switches to the unbiased masked policy gradient (Theorem 1); *masked, analytic V* adds the analytic recovery value (Proposition 6); and SafeExplorer adds the hard outcome gate (Equation (8)), with PPO (no recovery) anchoring the no-intervention end. Figure 1 ranks these variants by falls-to-success and Figure 4 (Appendix E.3) gives the complementary reward-vs-falls view. Each ingredient contributes a distinct improvement, attributed rung by rung below.

The first rung, the masked policy gradient (Theorem 1), is the foundation. Its biased alternative, Unmasked PG, reaches competitive or higher reward on HALFCHEETAH and GO1, but pays 14×, 7×, and 4× more falls

than SafeExplorer on HalfCheetah, Ant, and Go1 (Table 8), and is ill-posed for deterministic recovery. Masking removes that bias, the prerequisite for the next two ingredients.

Removing the bias exposes a second problem: the unbiased gradient is sparse at the safe-region boundary, so the masked, learned-V variant alone regresses to 1982 reward on HalfCheetah, the lowest of any rung (Table 8). The analytic recovery value (Proposition 6) supplies dense, correct targets exactly there, lifting HalfCheetah reward to 3870, the largest single-ingredient gain on that environment (+1888).

The top rung adds the hard outcome gate (SafeExplorer), which dominates on safety in every environment and supplies the largest single-ingredient reward gains on Go1 and Ant (+1121 and +2683; Table 8), where the analytic value alone leaves reward flat; it is the only variant to reach the Ant success bar (2286; next best 1253), the strongest evidence for the imitation reading of $C_\theta(\tau)$ (Section 4.4).

In the λ_{compat} sweep (Table 9; HalfCheetah and Ant only, the default transferred to Go1 untested), HalfCheetah reward stays at 4768 to 5783 across $[10^{-3}, 1]$, peaking at 10^{-2}, but falls are minimized at the 10^{-3} default (47) and rise 8× within a decade (374 at 10^{-2}); we choose the default on falls, not reward. Appendix E reports the rest: robustness to hyperparameters and noise (Appendices E.5 to E.7, E.9 and E.10), the soft-gate variants (Appendix E.4), compute-scaling (Appendix E.8), the full ladder (Appendix E.3), and a failure-mode catalog (Appendix E.11).

9 Discussion and Limitations

When the method helps, and when it does not. SafeExplorer's advantage is smallest when PPO is already safe and recovery is reliable, and largest in the opposite regimes: weak recovery (Ant), dangerous unaided PPO (Go1). The hard outcome gate makes this work: a soft gate imitates failed recovery actions and degrades both metrics (Appendix E.4). We treat the recovery policy as a permanent fallback, not temporary scaffolding: on Go1 it stays mildly engaged ($\alpha(\theta, d) \approx 0.05$) even at 16M steps, and keeping it available is what bounds falls there (Appendix E.12); our claim is reduced training falls, not their elimination.

Assumptions behind the guarantees. The analytic value (Proposition 6) is exact only under deterministic dynamics and recovery. Under noise it becomes a single-sample estimate, unbiased up to critic error at re-entry, favorable in practice since π^{rec} is near-deterministic; a multi-sample extension is in Appendix A.5. The objective-gap bound (Theorem 4) is controlled by the unobservable out-of-region rate $\beta(\theta, d)$, tracked via the observable $\alpha(\theta, d)$ as a diagnostic (Appendix E.12). A certified observable bound remains open (a deterministic recovery makes the natural simulation-lemma bound vacuous; routes in Appendix A.7).

Generality of the compatibility regularizer. The gate is also what makes the regularizer general: without it L^{compat} is DAgger-style imitation (Proposition 10), so outcome-gated imitation should transfer to other teacher-intervention settings, human-in-the-loop RL (Spencer et al., 2020) or JSRL (Uchendu et al., 2023).

Beyond locomotion. That transfer argument is theoretical; our evaluation is locomotion-only, though the correction applies wherever a fixed recovery policy takes over at unsafe states (a scripted reset in manipulation, an emergency-stop in driving); validating these is future work. The method also presupposes a pre-trained recovery policy and a hand-designed distance $\mathcal{D}$, whose acquisition cost is excluded from the reported budgets (Appendices C and D).

10 Conclusion

We built SafeExplorer, an on-policy RL algorithm for safety-critical control that, as a single PPO update, reduces training-time falls by 26× to 233× across three locomotion environments while matching or beating standard PPO on reward.

These gains rest on a theoretical foundation for RL with an external recovery policy: a masked policy-gradient theorem unbiased for the mixed-policy return under any recovery, an objective-gap bound with an exact fixed point, an analytic recovery value, and an outcome-gated compatibility regularizer, all instantiated for *safe-region intervention* to minimize training falls.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 22–31, 2017.

Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, 2022.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:29304–29320, 2021.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *Proceedings of the 2019 European Control Conference (ECC)*, pp. 3420–3431, 2019.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1726–1734, 2017.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL `https://doi.org/10.1145/1553374.1553380`.

Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization. *Frontiers in Robotics and AI*, 9, 2022.

Jiawei Chiu, Jean-Pierre Sleiman, Mayank Mittal, Farbod Farshidian, and Marco Hutter. A collision-free mpc for whole-body dynamic locomotion and manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 4686–4693, 2022.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerík, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *CoRR*, abs/1801.08757, 2018.

Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 1407–1416, 2018.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, pp. 482–495, 2017.

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

Shixiang Gu, Tim Lillicrap, Zoubin Ghahramani, Richard E. Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Sehoon Ha, Joonho Lee, Michiel van de Panne, Zhaoming Xie, Wenhao Yu, and Majid Khadiv. Learning-based legged locomotion: State of the art and future perspectives. *The International Journal of Robotics Research*, 44(8):1396–1427, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/haarnoja18b.html`.

Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, G. Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. In *Robotics: Science and Systems XV*, 2019. doi: 10.15607/RSS.2019.XV.011.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. In Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (eds.), *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pp. 483–491. International Foundation for Autonomous Agents and Multiagent Systems, 2020. doi: 10.5555/3398761.3398821.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and Jo ao G. M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40:698 – 721, 2021.

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. *Journal of Machine Learning Research*, 25(285):1–6, 2024.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 652–661, 2016.

Sebastian Junges, N. Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-constrained reinforcement learning for mdps. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems*, 2015.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pp. 267–274, 2002.

Katie Kang, Paula Gradu, Jason J. Choi, Michael Janner, Claire J. Tomlin, and Sergey Levine. Lyapunov density models: Constraining distribution shift in learning-based control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10708–10733. PMLR, 2022.

Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2–3):193–208, 2002.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083, 2019. doi: 10.1109/ICRA.2019.8793698.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: rapid motor adaptation for legged robots. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh (eds.), *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi: 10.15607/RSS.2021.XVII.011.

Joonho Lee, Jemin Hwangbo, and Marco Hutter. Robust recovery controller for a quadrupedal robot using deep reinforcement learning. *CoRR*, abs/1901.07517, 2019.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5, 2020.

Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. In *Conference on Robot Learning (CoRL)*, 2024.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, 2018. doi: 10.1109/ICRA.2018.8463162.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. URL `http://jmlr.org/papers/v21/20-212.html`.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3878–3887. PMLR, 2018.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018. doi: 10.1109/ICRA.2018.8460528.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In Marc Toussaint, Antonio Bicchi, and Tucker Hermans (eds.), *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020. doi: 10.15607/RSS.2020.XVI.064.

Xun Pua and Majid Khadiv. Safe learning of locomotion skills from mpc. In *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pp. 459–466, 2024. doi: 10.1109/Humanoids58906.2024.10769799.

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`.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 91–100. PMLR, 2021.

William Saunders, Girish Sastry, Andreas Stuhlmüller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 2067–2069, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. 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. *CoRR*, abs/1707.06347, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Laura M. Smith, J. Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Learning and adapting agile locomotion skills by transferring experience. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023a. doi: 10.15607/RSS.2023.XIX.051.

Laura M. Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023b.

Laura M. Smith, Yunhao Cao, and Sergey Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10829–10836, 2024.

Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. Saute RL: Almost surely safe reinforcement learning using state augmentation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20423–20443. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/sootla22a.html`.

Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Sidd Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep RL with a safety critic. *CoRR*, abs/2010.14603, 2020.

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 (ICML)*, 2020.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In Hadas Kress-Gazit, Siddhartha S. Srinivasa, Tom Howard, and Nikolay Atanasov (eds.), *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. doi: 10.15607/RSS.2018.XIV.010.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. *CoRR*, abs/1805.11074, 2018.

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. doi: 10.1109/LRA.2021.3070252.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. Safe reinforcement learning via curriculum induction. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Akifumi Wachi, Xun Shen, and Yanan Sui. A survey of constraint formulations in safe reinforcement learning. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 8262–8271. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/913. URL `https://doi.org/10.24963/ijcai.2024/913`. Survey Track.

Nolan C. Wagener, Byron Boots, and Ching-An Cheng. Safe reinforcement learning using advantage-based intervention. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10630–10640. PMLR, 2021.

Tsung-Yen Yang, Tingnan Zhang, Linda Luu, Sehoon Ha, Jie Tan, and Wenhao Yu. Safe reinforcement learning for legged locomotion. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2454–2461, 2022. doi: 10.1109/IROS47612.2022.9982038.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality models for the MuJoCo physics engine, 2022. URL `http://github.com/google-deepmind/mujoco_menagerie`.

Weiye Zhao, Tairan He, Rui Chen, Tianhao Wei, and Changliu Liu. State-wise safe reinforcement learning: A survey. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 6814–6822. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/763. URL `https://doi.org/10.24963/ijcai.2023/763`. Survey Track.

A Full Proofs

A.1 Proof of Theorem 1 (unbiased gradient of $J^{\mathrm{mix}}(\theta)$)

Proof. Starting from the definition, $J^{\mathrm{mix}}(\theta) = \int p_\theta^{\mathrm{mix}}\, r(\tau)\, d\tau$. Under Assumption 1, differentiation and integration commute: $\nabla_\theta J^{\mathrm{mix}}(\theta) = \int \nabla_\theta p_\theta^{\mathrm{mix}}\, r(\tau)\, d\tau$. Factor p_θ^{mix} explicitly by partition membership of the visited states:

$$p_\theta^{\mathrm{mix}} = \underbrace{p(s_1)}_{\text{no } \theta} \cdot \underbrace{\prod_{t:s_t\in\mathcal{M}} \pi_\theta(a_t \mid s_t)}_{\text{only } \theta\text{-dependent factor}} \cdot \underbrace{\prod_{t:s_t\notin\mathcal{M}} \mu(a_t \mid s_t)}_{\text{no } \theta} \cdot \underbrace{\prod_{t=1}^{T} P(s_{t+1} \mid s_t, a_t)}_{\text{no } \theta}, \tag{10}$$

where $\mu(\cdot|s)$ denotes the (possibly Dirac) θ-independent action measure at $s \notin \mathcal{M}$. Equation (10) is an equality of measures on trajectory space and does *not* require μ to have a Lebesgue density. Deterministic, stochastic, point-mass at a hand-coded action, and pre-trained-policy measures are all covered. Only the $\mathcal{M}$-step factor depends on θ. By the product rule and the score-function identity applied factor-wise,

$$\nabla_\theta p_\theta^{\mathrm{mix}} = p_\theta^{\mathrm{mix}} \sum_{t:s_t\in\mathcal{M}} \nabla_\theta \log \pi_\theta(a_t \mid s_t). \tag{11}$$

No gradient of μ appears, because μ carries no θ-dependence; at no point do we evaluate log of a Dirac measure. Substituting back,

$$\nabla_\theta J^{\mathrm{mix}}(\theta) = \mathbb{E}_{\tau\sim p_\theta^{\mathrm{mix}}}\left[\Big(\sum_{t:s_t\in\mathcal{M}} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big)\, r(\tau)\right]. \qquad \square$$

Corollaries (Corollaries 2, 8 and 9). The proof never used the specific form of μ, only that μ carries no θ-dependence, so each corollary follows by simply instantiating the pair $(\mathcal{M}, \mu)$. The safe-region intervention case ($\mathcal{M} = \mathcal{R}$, $\mu = \pi^{\text{rec}}$) is the body of this paper. The Jump-Start RL case ($\mathcal{M} = \{(s,t) : t \geq h\}$ for a per-trajectory handoff step h, $\mu = \pi^{\text{teach}}$ for a fixed teacher) inherits the same factorization: the teacher is θ-independent, the handoff step is data-dependent but not θ-dependent, and the score function evaluates only on the student-controlled tail of each trajectory. The state-triggered shielded RL case ($\mathcal{M} = \{s : \Sigma \text{ does not engage at } s\}$, $\mu(a \mid s) = \delta(a - \sigma(s))$ for the shield's corrective action σ) inherits the same factorization with μ a Dirac measure that the proof's factor-level differentiation handles directly. The common payoff is the same in all three: the IS fix is ill-defined when μ has no density at the relevant states, whereas the factor-level proof never forms the ratio and so bypasses the need for a density on μ. We state the two non-safe-region specializations formally for reference.

Corollary 8 (Jump-Start RL (Uchendu et al., 2023))**.** *Let π^{teach} be a fixed teacher policy and h a per-trajectory handoff step. Take $\mathcal{M} = \{(s,t) : t \geq h\}$ (states reached after handoff; the timestep t is absorbed into the state, standard for finite-horizon MDPs, so $\mathcal{M}$ is a subset of the augmented state space) and $\mu = \pi^{\text{teach}}$. Then Theorem 1 gives an unbiased gradient of the student's contribution to the mixed return, where the score function is evaluated only on the student-controlled tail of each trajectory.*

Corollary 9 (State-triggered shielded RL)**.** *Let Σ be a* state-triggered *shield, whose engagement is decided by the state alone and which, at unsafe states, replaces the proposed action with a θ-independent corrective action $\sigma(s)$. Take $\mathcal{M} = \{s : \Sigma \text{ does not engage at } s\}$ and $\mu(a \mid s) = \delta(a - \sigma(s))$. Then Theorem 1 gives an unbiased gradient evaluated only at unshielded states, with no density required on the shield's correction.*

Action-triggered shields, as in Alshiekh et al. (2018), decide the override from the proposed action $a \sim \pi_\theta(\cdot \mid s)$: at states where only some actions are unsafe, the executed-action distribution mixes π_θ restricted to safe actions with the correction weighted by the θ-dependent probability of proposing an unsafe action, so it depends on θ and falls outside Theorem 1's hypothesis of a θ-independent μ.

Truncated importance sampling. One might hope that the singularity is only a problem for the naive IS estimator and that a more careful clipped estimator escapes it. It does not. Truncated-IS estimators such as V-trace (Espeholt et al., 2018) and Retrace (Munos et al., 2016) reach the mixed-policy gradient by clipping the importance ratio $p_\theta / p_\theta^{\text{mix}}$, but that ratio is still undefined at the deterministic-recovery steps where μ is a point mass. Clipping bounds variance downstream of the singularity rather than removing it. The factor-level proof above avoids the ratio entirely, which is exactly why it extends to the deterministic recovery policy our method uses.

A.2 Proof of Theorem 4 (objective-gap bound)

This subsection and the remainder of the appendix work in the infinite-horizon discounted convention fixed in Section 3, under which $V^{\pi_\theta^{\text{mix}}}$, $Q^{\pi_\theta^{\text{mix}}}$, $A^{\pi_\theta^{\text{mix}}}$, and the normalized discounted visitations ν^{π_θ}, $\nu^{\pi_\theta^{\text{mix}}}$ are stationary.

Proof. Theorem 1 shows we optimize J^{mix} rather than the deployment objective J, so the question is how far apart the two can be. We bound the gap by routing it through the states where the two policies actually differ. By the Performance Difference Lemma (Kakade & Langford, 2002),

$$J(\theta) - J^{\text{mix}}(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim \nu^{\pi_\theta},\, a \sim \pi_\theta(\cdot \mid s)} \left[A^{\pi_\theta^{\text{mix}}}(s,a) \right], \tag{12}$$

where $A^{\pi_\theta^{\text{mix}}}(s,a) = Q^{\pi_\theta^{\text{mix}}}(s,a) - V^{\pi_\theta^{\text{mix}}}(s)$. Split the expectation by the safe region:

$$J(\theta) - J^{\text{mix}}(\theta) = \frac{1}{1-\gamma} \left(\underbrace{\mathbb{E}_{s \in \mathcal{R}}[\cdot]}_{=0} + \mathbb{E}_{s \notin \mathcal{R}}[\cdot] \right).$$

On $s \in \mathcal{R}$, both π_θ and π_θ^{mix} sample from π_θ, so the inner expectation $\mathbb{E}_{a \sim \pi_\theta(\cdot \mid s)}[A^{\pi_\theta^{\text{mix}}}(s,a)] = 0$ by definition of $V^{\pi_\theta^{\text{mix}}}$. On $s \notin \mathcal{R}$, $|A^{\pi_\theta^{\text{mix}}}(s,a)| \leq |Q^{\pi_\theta^{\text{mix}}}(s,a)| + |V^{\pi_\theta^{\text{mix}}}(s)| \leq 2r_{\max}/(1-\gamma)$ since both $|Q|$ and $|V|$ are

bounded by $r_{\max}/(1-\gamma)$. Therefore

$$|J(\theta) - J^{\text{mix}}(\theta)| \leq \frac{1}{1-\gamma} \cdot \frac{2r_{\max}}{1-\gamma} \cdot \Pr_{s\sim\nu^{\pi_\theta}}[s \notin \mathcal{R}] = \frac{2r_{\max}}{(1-\gamma)^2}\,\beta(\theta, d). \qquad \square$$

Bound on the out-of-region rate $\beta(\theta, d)$ (Section 4.2). The gap above scales with $\beta(\theta, d)$, so the bound is only useful once $\beta(\theta, d)$ itself is controlled. Section 4.2 uses $\beta(\theta, d) \leq \eta\gamma/(1-\gamma) \leq \eta/(1-\gamma)$, which we now derive from the per-step invariance slack, under the additional hypothesis that episodes start inside the safe region, supp $p(s_1) \subseteq \mathcal{R}(d)$ (otherwise the initial step alone contributes $(1-\gamma)\Pr[s_1 \notin \mathcal{R}]$ to $\beta(\theta, d)$ even with $\eta = 0$). All three of our environments satisfy the hypothesis: episodes reset to an upright pose inside the region. Starting from a reachable initial state $s_0 \in \mathcal{R}$ (this derivation counts steps from zero, so s_0 here is the s_1 of the trajectory notation), Assumption 2 gives $\Pr[s_0, \ldots, s_t \in \mathcal{R}] \geq (1-\eta)^t$. Since $\{s_t \in \mathcal{R}\} \supseteq \{s_0, \ldots, s_t \in \mathcal{R}\}$, the marginal in-region probability dominates the all-stay probability, $\Pr[s_t \in \mathcal{R}] \geq (1-\eta)^t$, hence $\Pr[s_t \notin \mathcal{R}] \leq 1 - (1-\eta)^t$. With the discounted state-visitation $\beta(\theta, d) = (1-\gamma)\sum_{t\geq 0}\gamma^t \Pr[s_t \notin \mathcal{R}]$,

$$\beta(\theta, d) \leq (1-\gamma)\sum_{t\geq 0}\gamma^t\big(1-(1-\eta)^t\big) = \frac{\eta\gamma}{1-\gamma(1-\eta)} \leq \frac{\eta\gamma}{1-\gamma} \leq \frac{\eta}{1-\gamma},$$

the last step using $1-\gamma(1-\eta) \geq 1-\gamma$ and $\gamma \leq 1$.

Train-to-deploy reward gap (footnote in Section 4.3). The training-signal convention of Proposition 6 (zero per-step reward during recovery, one-time terminal reward r_{term}) and the unmodified task reward differ only at states outside $\mathcal{R}$ (and through the one-time r_{term}). Bounding the per-step discrepancy by $r_{\max}+|r_{\text{term}}|$ on those steps and summing the discounted visitation under π_θ, the returns the two conventions induce under π_θ differ by at most $(r_{\max} + |r_{\text{term}}|)\,\beta(\theta, d)/(1-\gamma)$, so the train-to-deploy chain is controlled by the same $\beta(\theta, d)$ as Theorem 4.

A.3 Proof of Corollary 5 (conditional fixed point)

Proof. The gap bound of Theorem 4 leaves $\beta(\theta, d)$ free, so the natural endpoint of the analysis is the case that drives $\beta(\theta, d)$ to zero. The corollary splits into two readings of when that happens.

Reading (R-A): If $\mathcal{R}(d_{\max})$ covers the reachable state space, then $\Pr[s \notin \mathcal{R}(d_{\max})] = 0$ under *any* action distribution, including π_θ alone. Hence $\beta(\theta, d_{\max}) = 0$, and Equation (5) gives $J^{\text{mix}}(\theta) = J(\theta)$ exactly. Furthermore, π_θ^{mix} is identical to π_θ on the reachable support of $p_\theta(\tau)$ (recovery is never engaged).

Reading (R-B): If $\mathcal{R}(d_{\max})$ is strictly contained in the reachable state space, the theory no longer forces π_θ to remain in $\mathcal{R}$, so its invariance slack $\eta_\star := \eta(\theta, d_{\max}) \geq 0$ (Assumption 2) need not be zero. Substituting $\beta \leq \eta_\star/(1-\gamma)$ (derived above under the initial-state hypothesis supp $p(s_1) \subseteq \mathcal{R}$) into Equation (5) gives the upper bound $|J(\theta) - J^{\text{mix}}(\theta)| \leq 2r_{\max}\eta_\star/(1-\gamma)^3$, which, unlike (R-A), is not pinned to zero. Whether $\eta_\star$ (and hence the gap) is small is environment-dependent: Appendix E.12 finds $\eta_\star \to 0$ on HalfCheetah and Ant, where the gap closes to seed noise, and $\eta_\star > 0$ on Go1, where a proportional residual remains. $\square$

A.4 Proof of Proposition 6 (analytic recovery value)

Proof. The gradient and gap results above treat $V^{\pi_\theta^{\text{mix}}}$ as given; what remains is to compute it cheaply on the recovery segments themselves, where the trajectory is no longer under the agent's control. By the definition of $V^{\pi_\theta^{\text{mix}}}$, $V^{\pi_\theta^{\text{mix}}}(s_t) = \mathbb{E}_{\tau\sim p_\theta^{\text{mix}}(\tau)|s_t}\left[\sum_{j=0}^{\infty}\gamma^j r_{t+j}\right]$. Under deterministic dynamics and deterministic recovery, the trajectory starting from a recovery-triggering state s_t is fully determined for the duration of the recovery segment $[t, t+k]$. The expectation collapses. Two cases:

Recovery success ($s_{t+k} \in \mathcal{R}$): the realized return over the segment is $G_{t,k} = \sum_{j=0}^{k-1}\gamma^j r_{t+j}$. From s_{t+k} the agent resumes under π_θ^{mix} with value $V^{\pi_\theta^{\text{mix}}}(s_{t+k})$. Hence $V^{\pi_\theta^{\text{mix}}}(s_t) = G_{t,k} + \gamma^k V^{\pi_\theta^{\text{mix}}}(s_{t+k})$.

Recovery failure (terminal): the realized return is $G_{t,k}$ over the k deterministic steps, the episode ends, no further reward accrues. Hence $V^{\pi_\theta^{\text{mix}}}(s_t) = G_{t,k}$.

Combining gives Equation (6). Our implementation instantiates this general statement under a chosen reward: zero per-step reward during recovery and a one-time terminal reward r_{term} accrued at the final segment step $(t+k-1)$ on failure. The terminal penalty is thus the only nonzero contribution to the segment return $G_{t,k} = \sum_{j=0}^{k-1} \gamma^j r_{t+j}$, which is therefore 0 on success and $\gamma^{k-1} r_{\text{term}}$ on failure, matching Equation (6). □

A.5 Analytic value under stochastic dynamics

The closed form just derived rests on the determinism assumption, so it is worth asking what survives when that assumption is relaxed. Proposition 6 assumes deterministic P and π^{rec}. Under process noise or stochastic recovery, the realized $\gamma^k V_\theta(s_{t+k})$ is no longer equal to $V^{\pi_\theta^{\text{mix}}}(s_t)$. It becomes a single Monte-Carlo realization drawn from the distribution over post-trigger trajectories, whose variance grows with the noise of P and π^{rec} accumulated over the segment. Two natural extensions preserve the spirit of bypassing critic bootstrapping across recovery. (i) A multi-sample MC target: re-roll the recovery M times from s_t (where the simulator supports state-resets) and average $G_{t,k_m} + \gamma^{k_m} V_\theta(s_{t+k_m})$ across the M realizations. This is an unbiased estimator of $V^{\pi_\theta^{\text{mix}}}(s_t)$ whose variance falls as $1/M$. (ii) The single-sample plug-in we already use is an unbiased one-sample MC estimate of $V^{\pi_\theta^{\text{mix}}}(s_t)$. Its bias is zero in expectation, but its single-realization noise reaches downstream GAE. The bias-variance trade-off is favorable in our setting because π^{rec} is approximately deterministic in practice (MPC, greedy SAC), so the per-segment variance is small. A formal analysis under controlled stochasticity, and the regime where (i) is worth the extra simulator calls, is left to follow-up.

A.6 Proof of Proposition 10

The recovery segments that supplied the analytic value also supply state-action pairs for the compatibility regularizer, and identifying its limiting cases shows that this loss is not an ad-hoc addition but a familiar imitation objective in disguise.

Proposition 10 (Special cases of the compatibility regularizer)**.** *Let π^{rec} be deterministic, so $a_t^{\text{rec}} = \pi^{\text{rec}}(s_t)$ at recovery-controlled states, and consider the gate-set choice $\sigma_k \equiv 1$ for all segments.*

(a) ***Behavioral cloning on the recovery dataset.*** *Under $\sigma_k \equiv 1$,*

$$L_{\sigma\equiv 1}^{\text{compat}}(\theta) = -\frac{\lambda_{\text{compat}}}{N_{\text{rec}}} \sum_{t: s_t \notin \mathcal{R}} \log \pi_\theta(\pi^{\text{rec}}(s_t) \mid s_t)\,, \tag{13}$$

the (scaled) negative log-likelihood of π_θ on the dataset $\mathcal{B} = \{(s_t, \pi^{\text{rec}}(s_t)) : s_t \notin \mathcal{R}\}$ of recovery-controlled state-action pairs collected from rollouts of π_θ^{mix}. Equivalently, $L_{\sigma\equiv 1}^{\text{compat}}$ is the per-step behavioral-cloning objective applied to π^{rec}'s actions at the states π_θ^{mix} visits during recovery.

(b) ***A region-restricted DAgger aggregation step.*** *The dataset $\mathcal{B}$ is collected as in one iteration of DAgger (Ross et al., 2011) with the teacher queried only on the intervention set: states are visited under the mixed roll-out policy π_θ^{mix} (predominantly student-induced up to the safe-region boundary), and labels are the controller π^{rec}'s actions at the recovery-controlled states only, whereas DAgger labels every visited state. Minimizing $L_{\sigma\equiv 1}^{\text{compat}}$ as the per-iteration training step then reproduces a region-restricted analogue of DAgger's update with π^{rec} in the teacher role and π_θ as the student.*

(c) ***Outcome gate as a designer filter.*** *The general gate $\sigma_k \in \{0, 1\}$ restricts $\mathcal{B}$ to recovery segments that re-entered $\mathcal{R}$, discarding the rest. JSRL-style mixed-policy data collection (Uchendu et al., 2023) produces the same mixed-policy roll-out distribution but does not include an imitation loss. L^{compat} with $\sigma_k \equiv 1$ supplies the analogous student-imitation step that the JSRL algorithm itself omits. What is new in the case $\sigma_k \not\equiv 1$ is gating imitation of an* external recovery controller *by the* realized success of its multi-step segment*, as opposed to gating by value estimates of one's own past actions (Oh et al., 2018) or of demonstrations (Nair et al., 2018).*

Proof. **(a)** From Equation (9) with $w_t = \sigma_{\mathrm{seg}(t)}$, where $\mathrm{seg}(t)$ is the index of the recovery segment containing t (Section 4.4), and the assumption $\sigma_k \equiv 1$,

$$L^{\mathrm{compat}}_{\sigma\equiv 1}(\theta) = -\frac{\lambda_{\mathrm{compat}}}{N_{\mathrm{rec}}} \sum_{t:s_t\notin\mathcal{R}} 1 \cdot \log \pi_\theta(a^{\mathrm{rec}}_t \mid s_t) = -\frac{\lambda_{\mathrm{compat}}}{N_{\mathrm{rec}}} \sum_{t:s_t\notin\mathcal{R}} \log \pi_\theta(\pi^{\mathrm{rec}}(s_t) \mid s_t)\,,$$

using $a^{\mathrm{rec}}_t = \pi^{\mathrm{rec}}(s_t)$ for deterministic π^{rec}. This is, up to the constant scale $\lambda_{\mathrm{compat}}/N_{\mathrm{rec}}$, the empirical negative log-likelihood of π_θ on the dataset $\mathcal{B} = \{(s_t, \pi^{\mathrm{rec}}(s_t)) : s_t \notin \mathcal{R}\}$, the (population-level) maximum-likelihood / behavioral-cloning objective for π_θ on $\mathcal{B}$.

(b) The dataset $\mathcal{B}$ is collected by rolling out $\pi^{\mathrm{mix}}_\theta$, retaining the state-action pairs at the unsafe-region timesteps. Under $\pi^{\mathrm{mix}}_\theta$, π_θ controls the rollout up to the first exit from $\mathcal{R}$ (so the visited boundary state s_t has π_θ-induced distribution), and π^{rec} provides actions thereafter. This matches the data-collection rule of a single DAgger iteration with π^{rec} as the teacher and π_θ as the student, states visited under the mixed roll-out policy and labels from the teacher, except that the teacher is queried only at the recovery-controlled states rather than at every visited state as in Ross et al. (2011). Minimizing $L^{\mathrm{compat}}_{\sigma\equiv 1}$ as the per-iteration update therefore reproduces a region-restricted analogue of the DAgger aggregation step.

(c) For general $\sigma_k \in \{0, 1\}$, the regularizer is

$$L^{\mathrm{compat}}(\theta) = -\frac{\lambda_{\mathrm{compat}}}{N_{\mathrm{rec}}} \sum_k \sigma_k \sum_{t:s_t\notin\mathcal{R},\, \mathrm{seg}(t)=k} \log \pi_\theta(\pi^{\mathrm{rec}}(s_t) \mid s_t)\,,$$

i.e. the behavioral-cloning objective restricted to the sub-dataset $\mathcal{B}_{\mathrm{succ}} = \{(s_t, \pi^{\mathrm{rec}}(s_t)) : s_t \notin \mathcal{R},\, \sigma_{\mathrm{seg}(t)} = 1\}$ collected from segments that re-entered $\mathcal{R}$. JSRL (Uchendu et al., 2023) likewise rolls out a mixed teacher-then-student policy but trains the student only by reinforcement on the student-controlled tail. It does not include a teacher-imitation step. The $\sigma_k \equiv 1$ specialization of L^{compat} is the analogous imitation loss JSRL omits. In the $\sigma_k \not\equiv 1$ case, the new element is the gating criterion and target: imitation of an external recovery controller gated by the realized success of its multi-step segment, rather than by value estimates of the agent's own past actions (Oh et al., 2018) or of demonstrations (Nair et al., 2018). □

A.7 Sketch: why an observable upper bound on β is hard in continuous actions

The bound on $\beta(\theta, d)$ above is stated in terms of the per-step invariance slack η, which is not directly measurable; a bound in terms of an observable quantity would be far more useful, and we close by explaining why that is hard. A natural attempt is to bound β by the observable recovery rate α via a simulation-lemma argument: $\beta = \Pr_{\nu^{\pi_\theta}}[s \notin \mathcal{R}] \le \alpha + \mathrm{TV}(\nu^{\pi_\theta}, \nu^{\pi^{\mathrm{mix}}_\theta})$. Here $\mathrm{TV}(\cdot,\cdot)$ denotes total variation (TV) distance. The TV term can in turn be bounded by the expected per-step TV between π_θ and $\pi^{\mathrm{mix}}_\theta$ via the standard simulation lemma (Kearns et al., 2002). But on unsafe states $\pi^{\mathrm{mix}}_\theta$ collapses to a Dirac (deterministic recovery) while π_θ remains absolutely continuous. The per-step TV at unsafe states is 1 pointwise, so the simulation-lemma term contributes a full $1/(1-\gamma) > 1$, and the bound $\beta \le \alpha + 1/(1-\gamma)$ exceeds 1 and is therefore vacuous (since $\beta \le 1$ trivially). Routes that may produce a non-vacuous bound include (i) replacing the deterministic recovery with a smoothed (LSE / Gaussian-mixture) variant whose TV with π_θ is finite, (ii) Wasserstein-action-gap bounds under known dynamics Lipschitz constants, and (iii) periodic offline evaluation of π_θ alone. We sketch (i) below and leave a complete treatment to follow-up work.

Sketch via smoothed recovery. Replace $\pi^{\mathrm{rec}}(\cdot|s) = \delta(a - \bar{a}(s))$ with $\tilde{\pi}_{\mathrm{rec}}(\cdot|s) = \mathcal{N}(\bar{a}(s), \sigma^2_{\mathrm{rec}} I)$ for small σ_{rec}. Then both $\pi_\theta(\cdot|s)$ and $\tilde{\pi}_{\mathrm{rec}}(\cdot|s)$ are absolutely continuous and, when the scales are matched ($\sigma_{\mathrm{rec}} \approx \sigma_\theta(s)$, with σ_θ the main policy's standard deviation), the Gaussian TV bound gives $\mathrm{TV}(\pi_\theta, \tilde{\pi}_{\mathrm{rec}}) \le \mathrm{const} \cdot \|\mu_\theta(s) - \bar{a}(s)\|/\sigma_{\mathrm{rec}}$ at unsafe states up to a scale-mismatch term (with mismatched covariances the TV is bounded away from zero even at equal means), so the per-step TV is integrable under these stated conditions. The constant, however, grows with the action dimension, so even this smoothed bound loosens in high dimensions, the same mechanism by which the importance weights collapse on the 12-dimensional Go1 action space (Appendix E.13). Moreover, the TV to a fixed-scale π_θ approaches one as $\sigma_{\mathrm{rec}} \to 0$, so the smoothing scale trades fidelity to the deployed deterministic recovery against tightness of the bound; the sketch therefore yields an observable upper-bound proxy for β only under the matched-scale conditions above, not a general bound.

Algorithm 1 SafeExplorer: PPO with safe-step masking, analytic recovery V, and hard $C_\theta(\tau)$.

Require: Main policy π_θ, recovery policy π^{rec}, value function V_θ, safe-region radius schedule d_{iter}, total updates N, rollout length T_{ro}, num envs E, compatibility coef λ_{compat}, learning rate η_{lr}, terminal reward convention r_{term} (Appendix C.2).

for update $= 1, \ldots, N$ **do**
  $d \leftarrow d_{\text{iter}}(\text{update})$
  Rollout phase: for T_{ro} steps in each of E parallel envs,
    if $s_t \in \mathcal{R}(d)$: $a_t \sim \pi_\theta(\cdot|s_t)$, store $(s_t, a_t, r_t, \log \pi_\theta(a_t|s_t), V_\theta(s_t))$
    if $s_t \notin \mathcal{R}(d)$: $a_t = \pi^{\text{rec}}(s_t)$, $\texttt{is_rec}_t \leftarrow 1$
      store $(s_t, a_t, r_t, \log \pi_\theta(a_t|s_t), V_\theta(s_t))$ ▷ $\log \pi_\theta(a_t^{\text{rec}}|s_t)$ feeds L^{compat} only; never enters the PPO ratio (safe-step mask).
  Analytic-V overwrite (Proposition 6): for each recovery segment $[t_k^{\text{start}}, t_k^{\text{end}}]$ of length ℓ_k,
    if segment ended in re-entry: $V_\theta(s_{t_k^{\text{start}}}) \leftarrow \gamma^{\ell_k} V_\theta(s_{t_k^{\text{end}}})$ ▷ the segment return $G_{t,k}$=0 under the zeroed-recovery-reward convention, Proposition 6
    if segment ended in termination: $V_\theta(s_{t_k^{\text{start}}}) \leftarrow \gamma^{\ell_k - 1} r_{\text{term}}$ ▷ the terminal reward; recovery steps are zeroed, so this is $G_{t,k}$ on failure
    if segment cut off by rollout truncation: treat as re-entry at the truncation boundary ▷ tentative success with a bootstrapped value, Section 4.4
  Advantage estimation: compute $\hat{A}^{\text{full}}$ via GAE on the overwritten value sequence; the value target is $\hat{R}_t = \hat{A}_t^{\text{full}} + V_\theta(s_t)$.
  Outcome gates: for each recovery segment k, set $\sigma_k \leftarrow \mathbb{1}$[segment ended in re-entry].
  for epoch $= 1, \ldots, K$ **do**
    for minibatch $\subset$ rollout **do**
      $T_{\mathcal{R}} \leftarrow \{t : \texttt{is_rec}_t = 0\}$ (safe-step mask)
      $L_{\text{safe}}^{\text{PPO}} \leftarrow -\frac{1}{|T_{\mathcal{R}}|} \sum_{t \in T_{\mathcal{R}}} \min(\rho_t \hat{A}_t^{\text{full}}, \text{clip}(\rho_t) \hat{A}_t^{\text{full}})$
      $L^V \leftarrow \frac{1}{T_{\text{ro}}} \sum_t (V_\theta(s_t) - \hat{R}_t)^2$ ▷ unmasked: recovery-trigger targets are variance-free, exact up to critic error at re-entry
      $L^{\text{compat}} \leftarrow -\frac{\lambda_{\text{compat}}}{N_{\text{rec}}} \sum_{t \notin \mathcal{R}} \sigma_{\text{seg}(t)} \log \pi_\theta(a_t^{\text{rec}}|s_t)$
      $L \leftarrow L_{\text{safe}}^{\text{PPO}} + c_v L^V + L^{\text{compat}} - c_e H[\pi_\theta]$
      $\theta \leftarrow \theta - \eta_{\text{lr}} \nabla_\theta L$
    end for
  end for
end for

B Algorithm Details

B.1 Full pseudocode

Algorithm 1 gives the complete SafeExplorer update. The per-environment curriculum grows the safe-region radius linearly, $d_{\text{iter}}(u) = d_0 + \frac{u-1}{N} d_{\max}$ at update u of N, with $d_0 = 0.01$ on HalfCheetah and Ant and $d_0 = 0.05$ on Go1, up to the per-environment $d_{\max}$ values in Table 4 (Appendix C).

B.2 Notation

Table 2 collects the symbols used in Algorithm 1 and the design rationale below.

B.3 Operational features of the safe-region intervention setup

The safe-region intervention mechanism produces a training-time setup with four operational features that simultaneously place it outside CMDP and pointwise-filter approaches: (i) the safety predicate is binary set membership $s \in \mathcal{R}$ rather than a scalar cost, so there is no Lagrangian dual to optimize; (ii) π^{rec} may be deterministic (MPC, greedy SAC), in which case it has no Lebesgue density and the IS correction is ill-defined; (iii) recovery takes control for multi-step segments rather than at a single step, so training rollouts

Table 2: Notation used in Algorithm 1 and throughout the paper.

$\pi_\theta(a \mid s)$	main policy, stochastic, θ-parameterized
$\pi^{\text{rec}}(a \mid s)$	recovery policy; no θ-dependence; arbitrary form
π_θ^{mix}	mixed policy (Section 3)
$\mathcal{R}$, d, $d_{\max}$	safe region, current boundary, per-environment safe-region maximum (tuned hyperparameter)
$p_\theta(\tau), p_\theta^{\text{mix}}(\tau)$	trajectory distributions under π_θ alone and the mixed policy
$J(\theta), J^{\text{mix}}(\theta)$	main-policy return (unobservable during training) and mixed-policy return
$\alpha(\theta, d)$	recovery rate (observable)
$\beta(\theta, d)$	main-policy out-of-region rate (unobservable during training)
$C_\theta(\tau)$	compatibility score $\prod_{t \notin \mathcal{R}} \pi_\theta(a_t^{\text{rec}} \mid s_t)$
$V^{\pi_\theta^{\text{mix}}}, Q^{\pi_\theta^{\text{mix}}}, A^{\pi_\theta^{\text{mix}}}$	value, Q, advantage under the mixed policy
ℓ_k	length of recovery segment k
$\text{seg}(t)$	index of the recovery segment containing step t
σ_k	outcome gate of segment k (1 iff it ends with re-entry into $\mathcal{R}$)
N_{rec}	number of recovery-controlled steps in the current minibatch (normalizer of L^{compat})
r_{term}	environment reward on the terminal step of a failed recovery segment; per-environment values in Appendix C.2

follow a genuine mixed-policy distribution rather than a one-step perturbation; (iv) the objective is the original main-policy return $J(\theta)$, not a constrained surrogate, so safety is enforced operationally rather than penalized in the reward. Each of the four method components in Section 4 responds to a specific feature in this list.

B.4 Soft outcome gate

In addition to the hard outcome gate of Section 4.4, we considered a signed, per-step preference based on whether the post-segment state was higher- or lower-value than the pre-segment state, with $k(t)$ the length of the segment containing t:

$$w_t^{\text{soft}} = \gamma^{k(t)} V_\theta(s_{t+k(t)}) - V_\theta(s_t), \quad \text{normalized to zero mean per minibatch.} \tag{14}$$

Empirically, the soft variant is unstable on HalfCheetah and Go1 (the soft-gate variants in Appendix E.4) because the regularizer fits failed-recovery actions, so we drop it from the headline comparison.

B.5 Random-denominator and value-loss target choices

Two further choices in Algorithm 1 warrant explanation: the denominator of the masked policy gradient and the set of timesteps the critic regresses on. The masked PG estimator divides by $|T_{\mathcal{R}}|$, a random variable that depends on the rollout. Early in training, when $\alpha(\theta, d)$ is high, $|T_{\mathcal{R}}|$ can be small: with $m = |T_{\mathcal{R}}|$ safe steps the masked mean has variance on the order of σ^2/m, so a small m inflates the per-update gradient variance, and the random denominator additionally introduces a small ratio bias, one of the standard departures noted after Corollary 3. We do not apply variance-control heuristics. The curriculum schedule (Section 6) keeps $|T_{\mathcal{R}}|$ above a working threshold within the first few updates on every environment we evaluate.

The value-loss default also depends on whether the analytic recovery value (Proposition 6) is in use:

- **Analytic V on:** train the critic on *all* timesteps. Recovery-state targets under Proposition 6 carry no Monte-Carlo variance and are exact over the segment; the only residual error is the critic's own error at the single re-entry state (failure-segment targets are exact), so unmasked is principled and gives the critic dense supervision.
- **Analytic V off:** prefer the masked value loss $L^V(\theta) = |T_{\mathcal{R}}|^{-1} \sum_{t: s_t \in \mathcal{R}} (V_\theta(s_t) - \hat{R}_t)^2$. Otherwise the critic regresses against bootstrapped values that themselves depend on the critic at unsafe states, creating a self-consistency loop.

Our experiments use the analytic-V variant by default. Section 8 ablates this choice.

B.6 Where the changes land in the rollout/update loop

The choices above touch a standard PPO loop at a few points; the list below states where each is implemented.

- **Rollout, recovery branch:** when $s_t \notin \mathcal{R}$, the recovery action is queried, the action is stored in the buffer, $\log \pi_\theta(a_t \mid s_t)$ is computed at the recovery action, and the step is flagged as recovery-controlled.
- **Analytic-V overwrite:** for each recovery segment, the segment endpoint is identified and the value at the triggering state is overwritten with $\gamma^\ell V_\theta(s_{t+\ell})$ on success or $\gamma^{\ell-1} r_{\text{term}}$ on failure, where ℓ is the segment length. This happens in the rollout-buffer pass, before GAE.
- **GAE** (standard, no modification): operates on the overwritten value sequence.
- **Compatibility loss:** computes $\log \pi_\theta(a_t^{\text{rec}} \mid s_t)$ at unsafe steps and applies the chosen gate (σ_k for the hard gate, the signed advantage w_t^{soft} for the soft variant). The resulting loss is added to the PPO surrogate before backpropagation.
- **Masked PG and value loss:** the safe-step indicator masks both the policy-gradient term and (in the learned-critic variant, without the analytic value) the value loss.

B.7 Baseline ports

The two recovery-based baselines in Table 1 are ports into the same on-policy loop as SafeExplorer: both run inside the identical PPO update, share the identical safe-region trigger $s \notin \mathcal{R}(d)$ with the same curriculum on d, and query the identical frozen SAC recovery policy. They differ from SafeExplorer only in how recovery-controlled transitions enter the rollout buffer.

- **Recovery RL port** (Thananjeyan et al., 2021): at recovery steps the buffer keeps the *task policy's* action, value, and log-probability, the action-relabeling data-handling rule of the original method; recovery-step rewards are zeroed in the learning signal, as for SafeExplorer. The original method treats the recovery as part of the environment dynamics, which is coherent for its off-policy Q-learning objective; the bias corrected in this paper arises only when that relabeling is carried into an on-policy policy-gradient update.
- **Safe Legged port** (Yang et al., 2022): the same relabeling, plus a reward of $r_t - 1$ stored at recovery-triggered steps, the recovery-penalty shaping component of the published method.

What is deliberately *not* ported: Recovery RL's learned safety critic and off-policy training, and the model-based switching criterion of Yang et al. (2022). Replacing both with the shared distance trigger isolates each method's data-handling rule under a matched on-policy setup with an identical trigger and identical recovery controller, which is what makes the comparison interpretable. No baseline-specific tuning was performed beyond the shared PPO configuration of Appendix C.

B.8 Pseudocode parameters

Table 3 gives the default values that instantiate Algorithm 1 as run in our experiments.

C Hyperparameters

C.1 Training hyperparameters (shared across all variants)

Every method in Table 1 except the two CMDP baselines is trained with the single PPO configuration of Table 4, so that any difference among them reflects the method rather than per-variant tuning. CPO and PPO-Lagrangian additionally carry the constraint hyperparameters their objective requires (Appendix C.3). The compatibility regularizer λ_{compat} is the only entry active for SafeExplorer alone. All remaining hyperparameters are shared. The per-environment $d_{\max}$ values were set by a qualitative criterion, large enough

Table 3: Default parameter values for Algorithm 1.

Symbol in Algorithm 1	Default value
N (total updates)	total_timesteps/(num_envs × num_steps)
E (parallel envs)	4
T_{ro} (rollout length)	2048
K (PPO epochs)	10
c_v (value coef)	0.5
c_e (entropy coef)	0
λ_{compat} (compat coef)	10^{-3}
ϵ (PPO clip)	0.2
γ, GAE λ	0.99, 0.95
r_{term} (terminal reward)	environment terminal-step reward (Appendix C.2)

that the recovery policy stops triggering under normal task operation as d approaches $d_{\max}$; no systematic search over $d_{\max}$ was run.

Table 4: PPO training hyperparameters for every variant in Table 1. Per-environment $d_{\max}$ values are in env-specific task-space units, not comparable across environments.

Parameter	**Value**
Optimizer	Adam (eps = 10^{-5})
Learning rate	3×10^{-4}
Anneal LR linearly to 0	yes
Number of parallel envs	4
Rollout length per env	2048
Batch size (num_envs × num_steps)	8192
Mini-batch size	batch size / 32
GAE λ	0.95
Discount factor γ	0.99
PPO ratio clipping ϵ	0.2
Update epochs per batch	10
Value loss coefficient c_v	0.5
Max gradient norm	0.5
Entropy coefficient c_e	0
Compatibility regularizer λ_{compat} (SafeExplorer only)	10^{-3} (see Appendix E.5 for sweep)
Initial safe-region radius d_0	0.01 (Go1: 0.05)
Per-env $d_{\max}$	HalfCheetah: 2.0; Ant: 0.4; Go1: 0.15
Per-step recovery reward in the learning signal	0 (zeroed; Section 4.3)
Terminal-step reward r_{term}	environment terminal reward (Appendix C.2)

C.2 Terminal-step reward r_{term}

Intermediate recovery-step rewards are zeroed in the learning signal (Section 4.3), but the terminal-step reward that the analytic-V failure branch propagates as r_{term} (Algorithm 1) is the environment's *own* reward on the terminating step, preserved in the buffer. Concretely: on HalfCheetah the terminal step includes an explicit -1 penalty; on Go1 the per-step reward is floored at zero and the healthy bonus vanishes on an unhealthy step, so $r_{\text{term}} = 0$ (a configurable termination-penalty flag exists but is 0 in all main-table runs); on Ant the constant healthy bonus of 1.0 is paid even on the terminal step, so its terminal reward is that bonus minus the step's control and contact costs.

C.3 CMDP baseline hyperparameters

Both CMDP baselines run on OmniSafe 0.5.x. The constraint cost is 1 on each unhealthy termination and 0 otherwise; time-limit truncations incur no cost, matching the fall-counting convention of Appendix D. The cost limit is 0.05. Shared solver settings: steps per epoch 2048, 10 update iterations, minibatch size 64, target KL 0.02, $\gamma = \gamma_{\text{cost}} = 0.99$, GAE $\lambda = \lambda_{\text{cost}} = 0.95$, actor and critic MLPs of two hidden layers of size 64 with tanh activation, learning rate 3×10^{-4} for both with linear decay, observation normalization on, reward normalization on, cost normalization off. PPO-Lagrangian additionally uses ratio clip 0.2, Lagrange multiplier initialized at 0.001, and multiplier learning rate 0.035. CPO uses the OmniSafe defaults for its trust-region settings plus the cost limit above. No baseline-specific tuning was performed beyond these settings.

C.4 Per-environment training budget

The one quantity that does vary across environments is the training budget, which we fix before any rollout begins and report in Table 5. The larger budget on Go1 reflects its 12-DoF action space and longer main-task time-to-converge; the frozen SAC recovery policy (Appendix D.3) does not enter this budget. Matching all environments at 16M, or repeating Go1 at 8M for a fully matched comparison, is left to follow-up work.

Table 5: Per-environment training budget and SafeExplorer wall-clock cost.

Environment	Total env steps	Wall-clock (4 CPU cores)
HalfCheetah	8×10^6	∼3.5 h
Ant	8×10^6	∼5 h
Go1	16×10^6	∼12 h

C.5 Network architecture

Within each run, the network that PPO optimizes is the standard CleanRL actor/critic: the actor and critic are independent MLPs with two hidden layers of size 64 and tanh activation. The actor outputs a mean vector and a learnable global $\log \sigma$ for a diagonal-Gaussian policy. Layer weights are initialized orthogonally. The critic's output layer uses gain 1.0 and the actor's mean output uses gain 0.01.

C.6 Curriculum schedule

The remaining quantity that changes during a run is the safe-region radius, which is not held fixed but annealed along the linear schedule of Section 5, interpolating $d_0 \to d_{\max}$ over the $N = \text{total_timesteps}/(\text{num_envs} \times \text{num_steps})$ training updates. Appendix E.7 ablates the schedule shape (linear / log / step / constant) on Ant; the headline results all use the linear schedule.

D Environment and Recovery-Policy Details

D.1 Environment summary

We evaluate on three MuJoCo locomotion environments that differ in dimensionality and in how the safe region is sensed. Table 6 lists, for each one, the degrees of freedom, the observation and action dimensions, the indicator that the recovery policy uses to decide whether the agent has left the safe region $\mathcal{R}$, and the termination predicate that defines a fall. The indicator is a function of torso or base height together with tilt to the nominal pose, the signal the recovery policy is trained to restore.

Termination, horizon, and fall counting. The HalfCheetah termination predicate is a custom addition; the stock Gymnasium HalfCheetah never terminates, which is why our fall counts on HalfCheetah are nonzero. All three environments run under a 1000-step episode time limit. The control timestep is 0.05 s on HalfCheetah and Ant (frame skip 5 at a 0.01 s simulation step) and 0.01 s on Go1 (frame skip 5 at a 0.002 s simulation step), so the horizons are 50 s and 10 s of simulated time respectively. A *fall* is an unhealthy termination as defined in Table 6; time-limit truncations are never counted as falls, in training and evaluation alike.

Table 6: Per-environment dimensions, the safe-region indicator the recovery policy uses, and the termination predicate that defines a fall.

Environment	DoF	Obs dim	Action dim	Safe-region indicator	Fall (termination) iff
HalfCheetah	6	17	6	torso height z + tilt to nominal	$z \notin [-0.5, 0.5]$ or $\lvert\theta_{\text{pitch}}\rvert > 1.57$
Ant	8	27	8	torso height z + tilt to nominal	$z \notin (0.3, 1.5)$ or non-finite state
Go1	12	49	12	base z + base tilt (roll, pitch) to nominal	$z \notin (0.22, 0.65)$ or $\lvert\text{roll}\rvert$ or $\lvert\text{pitch}\rvert > 60°$

D.2 Safe-region distance and curriculum

The membership test is $s \in \mathcal{R}(d)$ iff $D(s) \leq d$, with an environment-specific distance D computed from the base height and orientation:

- HalfCheetah: $D(s) = \|(z, \theta_{\text{pitch}})\|_2$ against the nominal $(0, 0)$, with z in meters and pitch in radians, unweighted.
- Ant: $D(s) = \|(z{-}0.75, \text{tilt})\|_2$, where $\text{tilt} = \frac{\pi}{180} \arccos(\text{axis}_z) \cdot \phi/\pi$ with $\phi = 2\arccos(q_w)$, a rescaled axis-angle deviation from upright (q_w is the scalar quaternion component and axis_z the vertical component of the rotation axis). This scaling strongly downweights orientation relative to height on Ant; it is an implementation choice that we state plainly and keep for fidelity with the trained runs.
- Go1: $D(s) = \|(z - 0.3, \text{roll}, \text{pitch})\|_2$, with z in meters and roll and pitch in radians.

Because each D mixes meters and (rescaled) radians in a single Euclidean norm, d and $d_{\max}$ carry the same mixed units and are therefore not comparable across environments, the reason Table 4 reports them as env-specific values. During training, d follows the linear curriculum $d = d_0 + \frac{u-1}{N} d_{\max}$ at update u of N updates (Algorithm 1), with the per-environment $(d_0, d_{\max})$ of Table 4.

D.3 Recovery-policy training

For each environment, the recovery policy π^{rec} is a separately pre-trained SAC actor, trained on the same MuJoCo dynamics in a recovery-flavored configuration: the forward and velocity-tracking task terms of the environment reward are zeroed, so the remaining reward favors surviving and staying near the nominal pose. The shared recipe is CleanRL SAC with twin 256-256 Q networks, a 256-256 tanh-squashed Gaussian actor, replay buffer of 10^6 transitions, $\gamma = 0.99$, batch size 256, and an auto-tuned entropy coefficient; recovery-training episodes are capped at 250 steps, and at deployment we use the actor's deterministic tanh-mean action. Recovery training is *not* reset-free. At every episode reset, the initial state is aggressively randomized to mimic the failure modes a learning task policy will produce.

Initial-state randomization (per env reset):

- **Joint positions:** sampled uniformly over the full mechanical joint range (or a large subset). For Ant, each joint has env-specific bounds (e.g., ± 0.52 to ± 1.23 rad from nominal). For Go1, all 12 joint positions are drawn from the full actuator range.
- **Base height (and optionally x, y):** base z is randomized over a range that includes low and near-fall configurations (e.g., $z \in [0.22, 0.45]$ for Go1).
- **Base orientation:** the base quaternion is randomized to produce varied roll, pitch, and yaw. For Ant, a random quaternion is generated and constrained so the forward vector has non-negative z ("face up"). For Go1, roll and pitch are drawn in $\pm\pi/10$ and yaw in $[0, 2\pi]$, again with a face-up constraint.
- **Velocities:** initialized with small noise around zero.

Per-environment training and cost. On HalfCheetah the recovery policy was trained on the stock environment with the forward reward disabled, for 5M environment steps; the deployed checkpoint is the

4M-step snapshot. This policy was trained with an earlier revision of the trainer, so we report its checkpoint provenance rather than a reconstructed reward. On Ant it was trained for 3M steps (2M-step snapshot deployed; policy learning rate 10^{-4}, Q learning rate 3×10^{-4}, target-smoothing $\tau = 0.002$) on the recovery-flavored environment, whose effective reward is the healthy bonus minus control-magnitude and contact costs, so the Ant recovery is torque-limited. On Go1 it was trained for 10M steps (5M-step snapshot deployed; $\tau = 0.005$, Q learning rate 10^{-3}, policy learning rate 3×10^{-4}) with the velocity-tracking terms zeroed; the effective reward is the healthy indicator minus quadratic pose, height, and joint-deviation costs, floored at zero. Recovery pre-training therefore costs 5M (HalfCheetah), 3M (Ant), and 10M (Go1) environment steps. These steps are not included in the training budgets of Table 5, and the falls incurred during recovery pre-training are not tracked. The cost is nonetheless bounded: at most 62.5% of a single 8M-step HalfCheetah or Ant training run (5M of 8M) and 62.5% of the 16M-step Go1 budget (10M of 16M), which bounds the sense in which the recovery policy is cheap to obtain.

Per-environment recovery quality. How well this training regime works depends on the environment. Empirically, the HalfCheetah and Go1 recoveries succeed at stabilization with high probability across the randomized initial states. The Ant recovery policy does not, producing the unreliable-recovery regime that is the most informative axis of variation in our results (see Section 7). We attribute this to the recovery policy's reward design rather than to undertraining: the control-magnitude penalty ($0.5\|a\|^2$ in the Ant reward) caps how much torque the policy can apply, and the harder 3D Ant stabilization needs exactly those aggressive corrections. Relaxing that penalty or adopting a stronger recovery class for Ant, and confirming the attribution with a longer-training control, is left to follow-up work.

Recovery quality on the operational distribution. The randomized-reset measurement above grades the recovery on states the designer samples, not on the states the learning policy actually drives it into (which, unlike the face-up-constrained reset distribution, can include face-down configurations), so the operational grade is, if anything, the harder of the two. To measure the latter, during a trained run we log every first exit from $\mathcal{R}$ (the state at which recovery is triggered), then replay the SAC recovery from each logged state and record the fraction that *survive*, meaning the recovery completes the segment without an episode termination. We do this for SafeExplorer and for the masked, analytic-V ablation rung of Section 8, which shares the recovery policy and the curriculum with SafeExplorer and differs from it only by the hard outcome gate $C_\theta(\tau)$ (Equation (8)). Table 7 aggregates all logged triggers of one representative run per cell. Two effects stand out. First, SafeExplorer leaves $\mathcal{R}$ between 2.1× and 4.2× less often, so it calls on the recovery far less. Second, on Ant, where the recovery policy is unreliable, the exits SafeExplorer produces survive 91.3% of the time against 70.4% for the ungated rung, a gap of 20.9 points. On HalfCheetah and Go1, where the recovery policy is reliable, both variants survive at essentially the same rate (about 100% and 97%). The Ant gap offers an operational reading of the +2683 reward the hard gate adds on Ant in Section 8. The gate biases the main policy toward the part of state space where the weak Ant recovery policy actually succeeds. The trigger counts in Table 7 place the binomial 95% confidence interval on each survival rate within ±0.5 points. These intervals reflect sampling over triggers within one run, not variation across training seeds, which we do not characterize here; the table is therefore a single-run illustration rather than a multi-seed estimate.

Table 7: Recovery quality on the *operational* trigger-state distribution. We replay the SAC recovery from every logged first exit from $\mathcal{R}$ of one representative trained run per cell. "Exits" counts the logged recovery triggers; "Survived" is the percentage of those exits the recovery completes without termination. The ungated rung is the masked, analytic-V variant of Section 8.

	Exits (triggers)		Survived (%)	
Environment	SafeExplorer	ungated	SafeExplorer	ungated
HalfCheetah	80,546	338,671	99.8	100.0
Ant	16,366	35,150	91.3	70.4
Go1	94,212	255,881	97.0	97.4

D.4 Go1 environment specification

The Go1 observation is 49-dimensional, in order: base linear velocity scaled by 2.0 (3), base angular velocity scaled by 0.25 (3), base height z (1), roll, pitch, yaw (3), commanded velocity scaled by 2.0 (3), joint positions minus their defaults (12), joint velocities scaled by 0.05 (12), and the previous action (12); the vector is clipped to ± 100. The action is 12 absolute joint-position targets driving MuJoCo position servos with proportional gain $k_p = 20$, joint damping 1.0, and an actuator force range of $\pm 23.7\,\mathrm{N\,m}$ ($\pm 35.55\,\mathrm{N\,m}$ at the knees). The commanded velocity (v_x, v_y, ω_z) is resampled once per episode, each component uniform in $[-0.5, 0.5]$. The reward is $4.0\,\exp(-\|v_{xy}^{\mathrm{cmd}} - v_{xy}\|^2/0.25) + 2.0\,\exp(-(\omega_z^{\mathrm{cmd}} - \omega_z)^2/0.25)$, plus a healthy bonus of 1.0 and a feet-air-time term of weight 1.0, minus costs with weights: torque 2×10^{-4}, vertical velocity 2.0, xy angular velocity 0.05, action rate 0.001, joint-limit violation 0.01, joint acceleration 1.25×10^{-9}, orientation 0.4, height $4.0\,|z - 0.3|$, and joint deviation 0.1; the total is floored at 0.

D.5 Custom XML modifications

Supporting the recovery training and value conventions above requires three changes to the environment definitions. Our custom MuJoCo XMLs extend the standard Gymnasium MuJoCo environments with: (i) explicit joint-range attributes used by the recovery policy's randomized-reset training (Appendix D.3), (ii) a configurable flag that zeros the training reward outside $\mathcal{R}$ (the learning-signal convention of Section 4.3; reported returns are unmodified), and (iii) for Go1, the `mujoco_menagerie` Unitree Go1 model (Zakka et al., 2022) with a custom 12-DoF position-actuator configuration and a velocity-tracking reward adapted from `legged_gym` (Rudin et al., 2021).

E Additional Results

Run-batch provenance. The sensitivity studies in this appendix come from dedicated sweep batches run independently of the Table 1 runs; each caption states its batch, operating point, and seed count (first 5 of 10 seeds, i.e. seeds 1 to 5, unless noted otherwise). Cumulative falls are heavy-tailed across seeds, so independent 5-seed batches at identical settings can differ noticeably in absolute falls while preserving the orderings the text relies on. Where a table's operating point differs from the body protocol of Section 6, specifically the wider radii $d_{\max} = 3.2$ on HalfCheetah and 0.56 on Ant used by the v10 sweep family, the caption says so; within-table comparisons remain matched.

E.1 Headline comparison: rliable interquartile-mean intervals

This appendix supplies the evidence behind every claim made in the body, ordered as a descent from the headline numbers to the design choices that produce them, the robustness checks that stress them, and the theory and baselines that justify them. We begin with the headline. Figure 2 gives the per-metric reward and total-falls intervals for the six comparison-set methods, complementing the falls-to-success headline (Figure 1). It shows rliable interquartile means with 95% stratified-bootstrap confidence intervals (Agarwal et al., 2021), drawn as ranked intervals so the falls and reward orderings read directly.

E.2 Per-environment learning curves

The interval plot summarizes the endpoints; the learning curves show how each method gets there. Figure 3 traces reward and cumulative falls across the full training run, the trajectory view of the headline endpoints just reported.

E.3 Additional ablations

Having established what the headline shows, we now ask which design choices produce it, starting from the component ablation. Section 8 analyzes that ablation in the body. Figure 4 gives its reward-vs-falls view, the trade-off complement to the falls-to-success ranking of Figure 1; here we also report the remaining

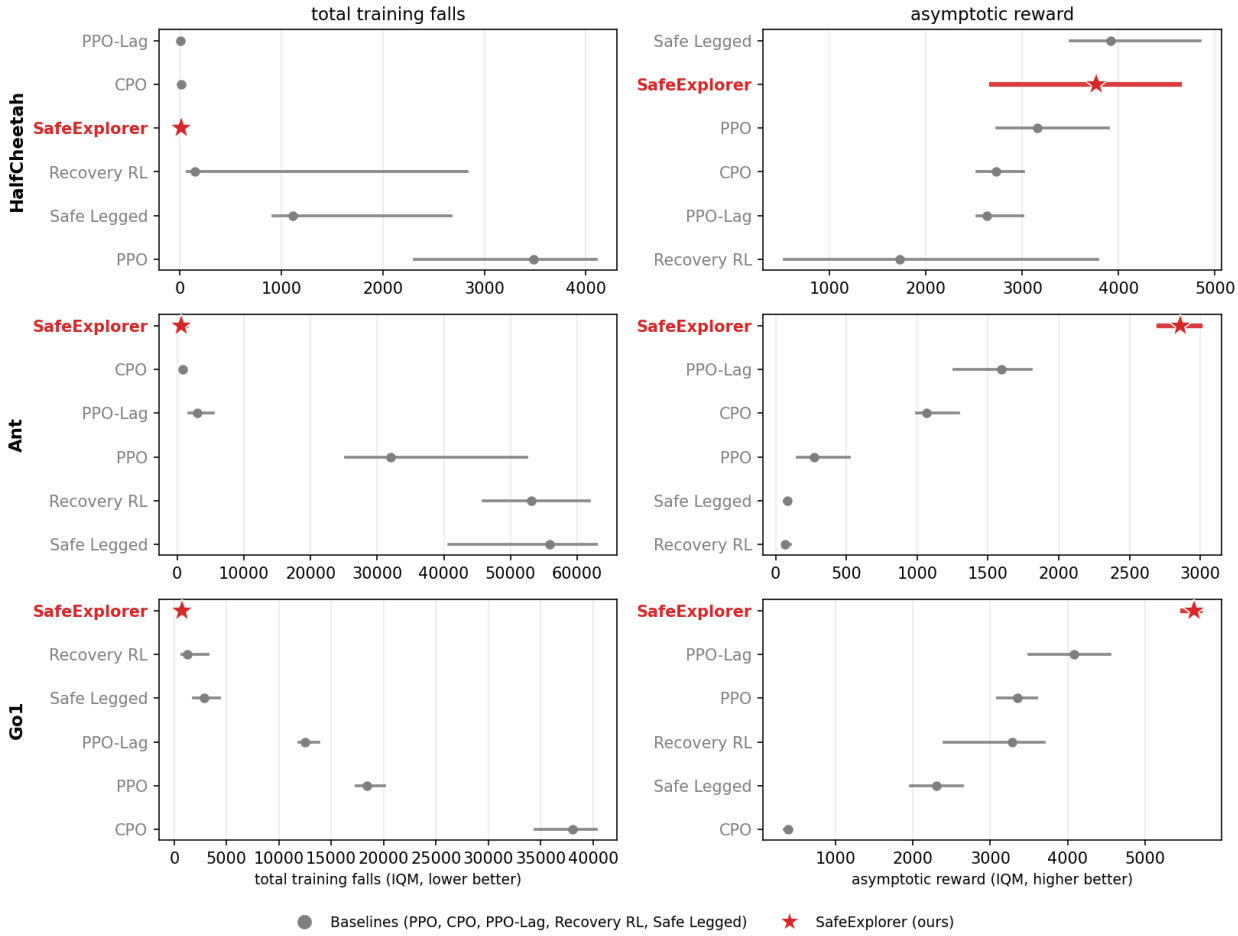

Figure 2: rliable interquartile-mean intervals of reward and total falls for the six methods in Table 1, with 95% stratified-bootstrap confidence intervals; a per-metric companion to the falls-to-success headline (Figure 1).

variants. The two threads opened here are picked up in turn below: the soft-gate variants (Appendix E.4) help marginally on Ant and hurt on HalfCheetah, and the λ_{compat} sweep is in Appendix E.5.

E.4 Soft compatibility-gate variants

The body ablation (Section 8, Figure 1) analyzes the clean ablation ladder Unmasked PG $\rightarrow$ masked learned V $\rightarrow$ analytic V $\rightarrow$ hard $C_\theta(\tau)$ (SafeExplorer); Table 8 tabulates its full numbers. The two soft-gate variants, a soft re-entry-advantage gate w_t^{soft} (Equation (14)) applied on the analytic-V model, and the same soft gate applied on a decomposed value, are negative results. They help marginally on Ant and hurt on HalfCheetah, and never approach SafeExplorer's safety. We report their full numbers here and omit them from the body ablation to keep it focused on variants that isolate a single design choice. Table 8 places them alongside the ladder rungs so the off $\rightarrow$ soft $\rightarrow$ hard progression is visible (same five-seed protocol as Table 1).

The mechanism behind the soft gate's failure is given in Section 4.4: on a failed recovery segment the soft gate still emits a negative-signed imitation pressure, teaching π_θ to *un*learn the recovery action at those states, a noisy signal that the hard gate removes by zeroing the term on failures.

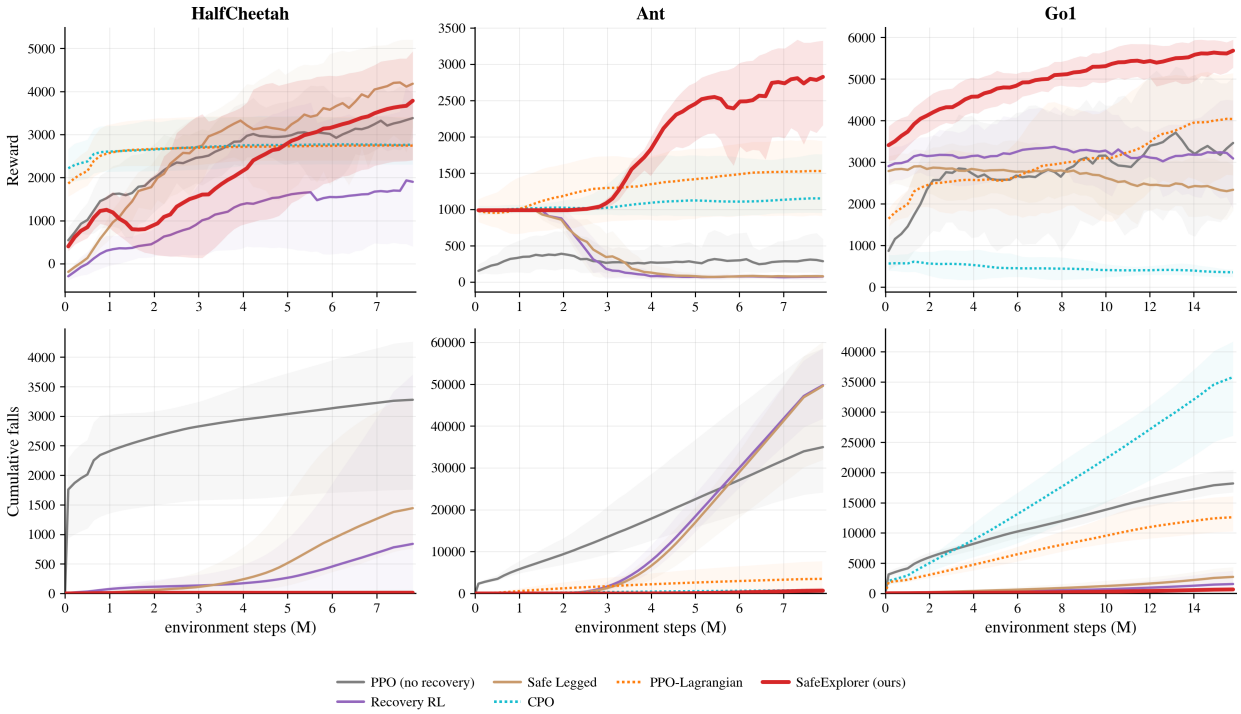

Figure 3: Per-environment learning curves: rows are episodic reward and cumulative training-time falls; columns are HalfCheetah, Ant, Go1. Solid = mean across seeds, shaded = [min, max] envelope; curves smoothed with a moving average for display. Recovery-rate curves are deferred to Appendix E.12.

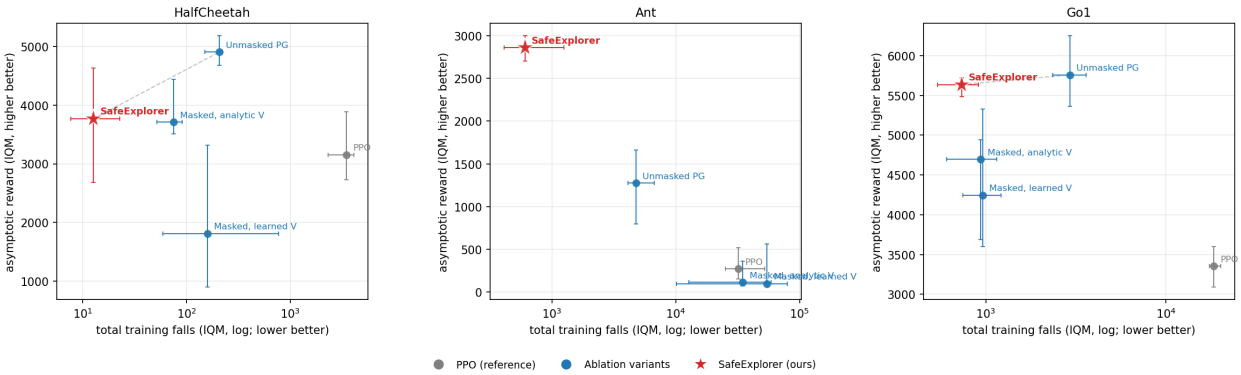

Figure 4: Ablation ladder on the reward-vs-falls plane: PPO and the four rungs, Unmasked PG → masked learned-V → masked analytic-V → SafeExplorer. Markers are rliable interquartile means with 95% stratified-bootstrap confidence intervals (Agarwal et al., 2021); total training falls on a log axis, upper-left is better (fewer falls, higher reward). The reward-vs-falls view of the variants Figure 1 ranks by falls-to-success; SafeExplorer (red star) is on the Pareto front in every environment.

E.5 λ_{compat} sensitivity (SafeExplorer)

With the gate fixed, the remaining knob on the regularizer is its weight. We sweep $\lambda_{\text{compat}} \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$ on HalfCheetah and Ant.

Table 8: Soft-gate ablations against the full ladder: Unmasked PG (the biased no-masking baseline of Section 8, added here so all rungs appear in one table), masked learned-V, analytic-V, and the hard gate (same protocol as Table 1). The soft gate is $w_t^{\text{soft}} = \gamma^k V(s_{t+k}) - V(s_t)$. **Bold** marks the best value per column.

	HalfCheetah		Ant		Go1	
Method	Reward	Falls	Reward	Falls	Reward	Falls
Unmasked PG	$\mathbf{4922 \pm 243}$	194 ± 40	1253 ± 427	5177 ± 1399	$\mathbf{5788 \pm 425}$	2936 ± 629
Masked, learned V	1982 ± 1058	306 ± 360	222 ± 275	50011 ± 30828	4374 ± 758	972 ± 206
Analytic V (no gate)	3870 ± 447	73 ± 19	174 ± 150	35481 ± 21851	4473 ± 628	926 ± 259
Soft $C_\theta(\tau)$ gate	2645 ± 829	1215 ± 1843	663 ± 323	11745 ± 18434	4142 ± 1471	2044 ± 2647
Soft $C_\theta(\tau)$, decomposed V	3569 ± 1074	721 ± 985	505 ± 319	8183 ± 8064	4374 ± 1191	1242 ± 472
SafeExplorer (hard $C_\theta(\tau)$)	3581 ± 971	$\mathbf{14 \pm 8}$	$\mathbf{2857 \pm 129}$	$\mathbf{754 \pm 463}$	5594 ± 177	$\mathbf{731 \pm 177}$

Table 9: Compatibility-regularizer weight sweep on HalfCheetah and Ant. Reward and falls are training-time (last-10% mean reward, cumulative falls); mean ± std over the first 5 of 10 seeds, dedicated sweep batch at the body operating point, run independently of the Table 1 runs. **Bold** marks the best value per column.

λ_{compat}	HalfCheetah reward	HalfCheetah falls	Ant reward	Ant falls
10^{-4}	2654 ± 1833	211 ± 266	1803 ± 1176	3557 ± 3795
10^{-3}	4768 ± 583	$\mathbf{47 \pm 41}$	2908 ± 157	428 ± 454
10^{-2}	$\mathbf{5783 \pm 481}$	374 ± 548	$\mathbf{3124 \pm 355}$	639 ± 517
10^{-1}	5288 ± 667	64 ± 57	2281 ± 894	$\mathbf{149 \pm 140}$
1.0	5536 ± 333	137 ± 151	1352 ± 1036	2798 ± 5658

The $\lambda_{\text{compat}} = 10^{-3}$ row is an independent replication of Table 1's configuration: it lands at HalfCheetah 4768 ± 583 / 47 ± 41 against Table 1's 3581 ± 971 / 14 ± 8, consistent within seed noise for a heavy-tailed falls count. The same runs also back Table 15 (HalfCheetah/Ant).

The optimum is environment-specific. On HalfCheetah the reward peaks sharply at $\lambda_{\text{compat}} = 10^{-2}$ (5783). On Ant the reward is within seed noise from $\lambda_{\text{compat}} = 10^{-3}$ to 10^{-2}, with 10^{-3} the safer choice. Both environments are robust within a $\sim 10\times$ band around the optimum.

The main results use a single $\lambda_{\text{compat}} = 10^{-3}$. SafeExplorer in Table 1 and Figure 3 fixes $\lambda_{\text{compat}} = 10^{-3}$ across all three environments rather than a per-environment optimum. We make this choice on *safety* grounds. In the sweep above, $\lambda_{\text{compat}} = 10^{-3}$ attains the lowest HalfCheetah training-fall count (47, against 374 at 10^{-2}) and a near-lowest count on Ant. A larger $\lambda_{\text{compat}} = 10^{-2}$ would raise HalfCheetah reward substantially (5783 vs 4768) and is marginally better on Ant, but it does so at roughly $8\times$ the HalfCheetah falls. We therefore keep the uniform 10^{-3} default so that the headline fall reductions are not traded away for extra falls, and we note 10^{-2} as the reward-optimal setting for HalfCheetah for practitioners who can tolerate the higher fall count.

E.6 Fixed-d ablation: the safe-region schedule vs. a constant d

With the gate and its weight fixed, the last training-time design choice is the safe-region radius, which SafeExplorer anneals on a curriculum. A natural counterfactual is to pick a smaller $d_{\max}$ and hold d constant there. Table 10 sweeps fixed $d \in \{0.25, 0.5, 0.75, 1.0\} \cdot d_{\max}$ on each of the three environments, with otherwise-identical SafeExplorer.

HalfCheetah. Every fixed-d cell pays $\sim$ 1,200 to 3,600 training falls and tops out near 2,500 to 3,600 reward, while the linear-curriculum reference reaches 3,581 reward at only 14 falls (an $85\times$ to $260\times$ falls advantage). No constant d matches the curriculum's reward-at-low-falls. The small-d cells stay safer but low-reward and the large-d cells climb in falls without gaining reward.

Table 10: Fixed-d sweep at the body operating point (SafeExplorer), dedicated sweep batch, mean $\pm$ std over the first 5 of 10 seeds on HalfCheetah/Ant and over all 5 seeds on Go1. d is held constant at the listed percentage of the per-env $d_{\max}$ (2.0 / 0.4 / 0.15) throughout training. Cells are `Reward / Falls`; **bold** marks the best reward and lowest falls per environment. Bottom row: linear-curriculum reference at the same $d_{\max}$, reproducing Table 1 (a separate run batch); the sweep's own linear arm at the same settings gives HalfCheetah 4773 ± 536 / 46 ± 41, Ant 2929 ± 157 / 419 ± 446, Go1 5613 ± 234 / 750 ± 231, an independent replication consistent within seed noise.

$d/d_{\max}$	HalfCheetah	Ant	Go1
25%	$3344 \pm 1642/1189 \pm 1274$	$990 \pm 2/\mathbf{5 \pm 4}$	$2898 \pm 147/1895 \pm 409$
50%	$2476 \pm 668/2623 \pm 664$	$1602 \pm 463/5522 \pm 1505$	$5423 \pm 119/\mathbf{397 \pm 33}$
75%	$\mathbf{3629 \pm 582}/3489 \pm 889$	$555 \pm 178/19941 \pm 4063$	$5572 \pm 179/888 \pm 372$
100%	$3535 \pm 1053/3632 \pm 641$	$565 \pm 301/21837 \pm 8376$	$\mathbf{5752 \pm 158}/1662 \pm 272$
Linear curr. (SafeExplorer, ref.)	$d_{\max}$=2.0: $3581 \pm 971/\mathbf{14 \pm 8}$	$d_{\max}$=0.4: $\mathbf{2857 \pm 129}/754 \pm 463$	$d_{\max}$=0.15: $5594 \pm 177/731 \pm 177$

Ant. The smallest fixed-d (p25, $d = 0.1$) is very safe (5 falls) but useless (990 reward). Larger constant d accumulates 5,500 to 21,800 falls without reaching the curriculum's reward. The linear curriculum gets 2857 reward at 754 falls, which no constant d matches.

Go1. On Go1 the comparison is closer, and the curriculum does *not* strictly dominate. A tuned constant d is competitive. Fixed-d-p50 (5423/397) attains essentially the curriculum's reward (5594) at *fewer* falls (397 vs 731), and p75 and p100 trade modest extra falls for similar reward. The curriculum's value on Go1 is therefore robustness to the choice of d, it reaches high reward at low falls without a per-environment d search, rather than a strict win over the best constant d.

Summary. On HalfCheetah and Ant no constant d matches the schedule's high reward at comparable safety. On Go1 a tuned fixed-d (p50) is competitive on both axes, so there the schedule's contribution is sparing the practitioner a per-environment d search rather than a strict dominance. The safe-region schedule is a supporting mechanism for the objective-gap bound (Theorem 4). The method's correctness rests on the unbiased masked gradient (Theorem 1), not on the schedule.

E.7 Curriculum-schedule sensitivity (SafeExplorer on Ant)

Table 11: Schedule shape on Ant SafeExplorer. Reward / Falls, mean $\pm$ std over the first 5 of 10 seeds; v10 sweep batch at the wider radius ($d_{\max} = 0.56$) rather than the body 0.4, so within-table comparisons are matched but absolute values differ from Table 1. **Bold** marks the best value per column; Linear and Log rewards tie within seed noise and are both bolded.

Schedule	Reward	Falls
Linear	$\mathbf{2888 \pm 351}$	$\mathbf{952 \pm 1121}$
Log	$\mathbf{2906 \pm 374}$	1086 ± 438
Step (4 jumps)	1537 ± 549	5998 ± 2788
Constant ($d = d_{\max}$)	294 ± 74	29855 ± 4747

Granting that some schedule beats a constant d, the next question is which one. Table 11 reports three alternatives against the linear default. Linear and log are tied within seed noise on every metric. The step schedule is worse on both, with lower reward and 6× more falls. Constant-d (no curriculum) collapses, accumulating 31× the falls of the linear curriculum at a tenth of its reward. This is consistent with the fixed-d ablation above (Appendix E.6), whose constant d=$d_{\max}$ cell shows the same qualitative collapse (at the body radius rather than this table's wider one), and with the prediction of Theorem 4 that the gap closes as β shrinks, which the curriculum drives.

E.8 Compute-scaling: 2× matched-budget on HalfCheetah and Ant

With the design choices settled, we turn to whether the chosen configuration holds up under conditions the body did not exercise, starting with more compute. We re-run PPO and SafeExplorer at 2× the main budget (8M → 16M) on HalfCheetah and Ant (Table 12; wider radii than the body protocol, see the caption). SafeExplorer's reward improves or holds with 2× compute (HalfCheetah 4027 → 5912, Ant 2888 → 2950) while its falls stay low. PPO's falls instead grow sharply (HalfCheetah ∼ 2.9×, Ant ∼ 2.7×), so the safety gap *widens* with budget rather than closing. Retuning $d_{\max}$ for longer horizons or a multi-stage curriculum is left to follow-up work.

Table 12: Compute-scaling: SafeExplorer vs PPO at the main budget (8M) and 2× (16M) on HalfCheetah and Ant. Reward / Falls, mean ± std over the first 5 of 10 seeds. All cells are the v10 batch at the wider radii ($d_{\max} = 3.2$ on HalfCheetah, 0.56 on Ant) rather than the body protocol; the 16M runs stretch the d-curriculum over the doubled budget. Table 1's body-radius values (HalfCheetah SafeExplorer 3581 ± 971 / 14 ± 8) therefore differ from the Main column here; see the preamble of this appendix. **Bold** marks the better method per environment and budget.

	Method	Main (8M)	2× (16M)
HalfCheetah	PPO	$3163 \pm 966/5266 \pm 2539$	$3590 \pm 1969/15112 \pm 8500$
	SafeExplorer	$\mathbf{4027 \pm 1239/248 \pm 281}$	$\mathbf{5912 \pm 881/217 \pm 340}$
Ant	PPO	$296 \pm 35/32369 \pm 5625$	$197 \pm 92/88900 \pm 35552$
	SafeExplorer	$\mathbf{2888 \pm 351/952 \pm 1121}$	$\mathbf{2950 \pm 419/3332 \pm 2620}$

E.9 Compatibility-regularizer normalization (global vs. per-segment)

The $C_\theta(\tau)$ loss in Equation (9) averages across all recovery-active timesteps in the current minibatch ("global"). An alternative "per-segment" normalization averages within each contiguous safe→unsafe→safe segment, then over segments, which prevents long unsafe excursions from dominating the loss. Table 13 reports both on HalfCheetah and Ant. Per-segment helps on HalfCheetah (higher reward, fewer falls) but hurts on Ant (lower reward, more falls). With no consistent winner across environments and the means within roughly one standard deviation, we keep `global` as the simpler default.

Table 13: Compat-norm: global vs. per-segment. Reward / Falls, mean ± std over the first 5 of 10 seeds; v10 batch at the wider radii ($d_{\max} = 3.2$ HalfCheetah, 0.56 Ant). **Bold** marks the arm better on both metrics per environment.

Env	global (default)	per-segment
HalfCheetah	$4027 \pm 1239/248 \pm 281$	$\mathbf{5030 \pm 664/48 \pm 36}$
Ant	$\mathbf{2888 \pm 351/952 \pm 1121}$	$2474 \pm 489/2625 \pm 1583$

E.10 Action-noise robustness (HalfCheetah SafeExplorer)

Table 14: Action-noise on HalfCheetah SafeExplorer at $\lambda_{\text{compat}} = 10^{-3}$. Reward / Falls, mean ± std over the first 5 of 10 seeds; v10 batch at the wider HalfCheetah radius ($d_{\max} = 3.2$). The $\sigma_a = 0$ arm is the same runs as the global arm of Table 13 and the Main SafeExplorer cell of Table 12.

σ_a	Reward	Falls
0.00	4027 ± 1239	248 ± 281
0.05	4482 ± 419	108 ± 73
0.10	2346 ± 964	546 ± 395

A second stress axis perturbs the actions themselves. We re-train SafeExplorer on HalfCheetah with Gaussian action noise of standard deviation $\sigma_a \in \{0.0, 0.05, 0.10\}$ added to every action (recovery and main) at the $\lambda_{\text{compat}} = 10^{-3}$ default (Table 14). Reward and falls stay within seed noise up to $\sigma_a = 0.05$, then degrade at $\sigma_a = 0.10$ (−42% reward, more than double the falls). An action-noise sweep at the HC-tuned $\lambda_{\text{compat}} = 10^{-2}$ is left to follow-up work. We also ran $\sigma_a = 0.20$ as a stress test. SafeExplorer collapses (1863 ± 542 reward, 1659 ± 1807 falls) when the injected noise is comparable in scale to the recovery's action magnitude, and we omit it from the table as a known out-of-regime failure mode.

E.11 Failure-mode catalog

The stress tests above probe how the chosen configuration degrades; this catalog records the qualitative failure each rejected variant exhibits, so the quantitative gaps in the ablations have a concrete behavioral reading.

- **PPO** on Go1: agent falls within first 200 steps; recovery is unavailable; episode terminates.
- **Analytic-V variant** on Ant: agent learns to balance briefly but never escapes the recovery policy's basin; α stays near 1 for the full training run.
- **Soft-gate variant** on HalfCheetah: regularizer fits failed-recovery actions; reward oscillates; final policy is unreliable.
- **SafeExplorer** on Ant: clean separation, α drops to near zero around update 1000, reward climbs steadily thereafter, falls plateau.

E.12 Empirical check of Theorem 4

We now turn from the method's empirical robustness to its theory, checking each of the paper's two theorems against the runs, starting with the objective-gap bound. Theorem 4 bounds the deployment gap $|J^{\mathrm{mix}}(\theta_k) - J(\theta_k)|$, between the during-training mixed-policy return and the return of π_θ deployed alone, by the main-policy out-of-region rate $\beta(\theta, d)$. The bound is a proportionality, not a vanishing guarantee. It predicts a small gap exactly when $\beta(\theta, d)$ is small, and Corollary 5 sharpens this to an exact fixed point only in the idealized $\beta(\theta, d) \to 0$ limit. We test that prediction directly. Every 4 PPO updates we run a recovery-disabled evaluation pass, averaging 5 episodes per pass, that samples from the same stochastic policy $\pi_\theta(\cdot|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta^2)$ used in training, the operative $J(\theta_k)$, and compare it to the mixed-policy rollout return $J^{\mathrm{mix}}(\theta_k)$. Because $\beta(\theta, d)$ is an out-of-region rate under π_θ alone and is not directly observable, we report alongside it the observable recovery rate α, which Section 4.2 identifies as a diagnostic for $\beta(\theta, d)$ rather than a certified upper bound. A deterministic pass using $\mu_\theta(s)$ alone is also logged as a noise-reduced deployment diagnostic but is not the quantity the theorem concerns.

Table 15: End-of-training values for SafeExplorer at the body operating point, first 5 of 10 seeds, over the last 20% of training; an independent batch shared with the $\lambda_{\mathrm{compat}} = 10^{-3}$ arm of Table 9 on HalfCheetah/Ant, with Go1 from the body-16M family. Columns: the mixed-policy return J^{mix}, the π_θ-only stochastic eval return J, their signed difference, the recovery rate α (observable diagnostic for $\beta(\theta, d)$), cumulative training-time falls, and per-episode eval-time falls under π_θ alone (recovery disabled).

Env	J^{mix}	J	$J^{\mathrm{mix}} - J$	α	Falls (train)	Falls (eval/ep)
HalfCheetah	4677 ± 542	4676 ± 552	0 ± 12	0.000	47 ± 41	0.00
Ant	2904 ± 145	2921 ± 151	-17 ± 25	0.000	428 ± 454	0.12
Go1	5576 ± 213	5112 ± 340	463 ± 150	0.044 ± 0.005	768 ± 241	0.36

The result follows the bound's structure precisely, and the key observation is that *the gap tracks* $\beta(\theta, d)$. On HalfCheetah and Ant the recovery rate falls to essentially zero ($\alpha = 0.000$ on both at end-of-training), and the gap closes to within seed noise of zero (0 ± 12 and -17 ± 25). These are the self-stable environments. Here π_θ holds the safe region unaided, so $\pi_\theta^{\mathrm{mix}} \equiv \pi_\theta$ along the trajectory, $\beta(\theta, d) \to 0$, and the fixed point of Corollary 5 is reached up to the invariance-slack residual $\eta_\star$ the corollary anticipates (the residuals here are statistically indistinguishable from zero; empirically it is the stochastic policy's entropy floor that keeps $\eta_\star$ from being exactly zero). On Go1 the policy does not become fully self-stable, because the quadruped's gait cannot stay inside the tight $\mathcal{R}(d_{\max})$ tube. The recovery rate therefore settles at a small positive floor ($\alpha = 0.044 \pm 0.005$) rather than at zero, and the gap settles at a correspondingly positive residual (463 ± 150, about 8% of J^{mix}) rather than vanishing. After the early-curriculum transient the gap collapses from its peak on HalfCheetah, stays within seed noise of zero on Ant in the final average despite a late upward drift in the plotted per-seed magnitude, and declines but stays positive on Go1 (Figure 5). On Go1 it falls

from ~ 1950 when the safe region is small to a small positive residual at 16M env-steps. The end-of-training gap and recovery rate in Table 15 make the proportionality explicit. The gap is near zero exactly where α is, and the only environment with a residual gap is the only one with a residual α.

This two-regime behavior is a stronger test of Theorem 4 than a uniform collapse to zero would be. The bound does not assert that the gap vanishes; it asserts that the gap is governed by $\beta(\theta, d)$. HalfCheetah and Ant realize the vanishing regime ($\beta(\theta, d) \to 0$), and Go1 realizes the residual regime ($\beta(\theta, d)$ bounded away from zero because Assumption 2 holds only approximately for a quadruped confined to a tight tube), with the observable α moving in lockstep with the gap in both. The sign of the gap is itself interpretable. On Go1, $J^{\text{mix}} - J > 0$ means that deploying π_θ alone forfeits the return that the recovery policy would otherwise secure, so the gap measures the policy's residual reliance on the recovery policy, which HalfCheetah and Ant drive to zero and Go1 reduces to about 8%.

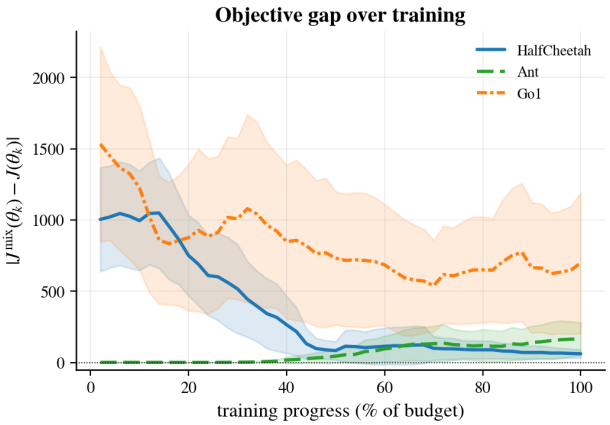

Figure 5: Objective gap over training for the SafeExplorer runs of Table 15: the curve is the seed-mean of the per-seed absolute gap $|J^{\text{mix}}(\theta_k) - J(\theta_k)|$, smoothed with a 9-point moving average for display; shaded = ± 1 std. Because the curve averages magnitudes while Table 15 reports the signed across-seed mean over the final 20% of training, the curve's end value can sit above the table's near-zero signed gap (as on Ant).

The absolute prefactor of Theorem 4 ($2r_{\max}/(1-\gamma)^2 \approx 2 \times 10^5$ for $r_{\max} = 10$, $\gamma = 0.99$) is loose, as is standard for performance-difference-lemma bounds, so we do not read the inequality quantitatively. Its operative content is the structural claim that any mechanism reducing $\beta(\theta, d)$, here the radius curriculum together with the compatibility regularizer, tightens the gap. Figure 5 confirms that claim across the full range of $\beta(\theta, d)$ our three environments realize. Eval-time falls under π_θ alone (recovery disabled) corroborate the reading. They are 0.00, 0.12, and 0.36 per episode on HalfCheetah, Ant, and Go1, ordered exactly as the gaps and the recovery rates are. Figure 6 plots the full recovery-rate trajectories that this diagnostic summarizes.

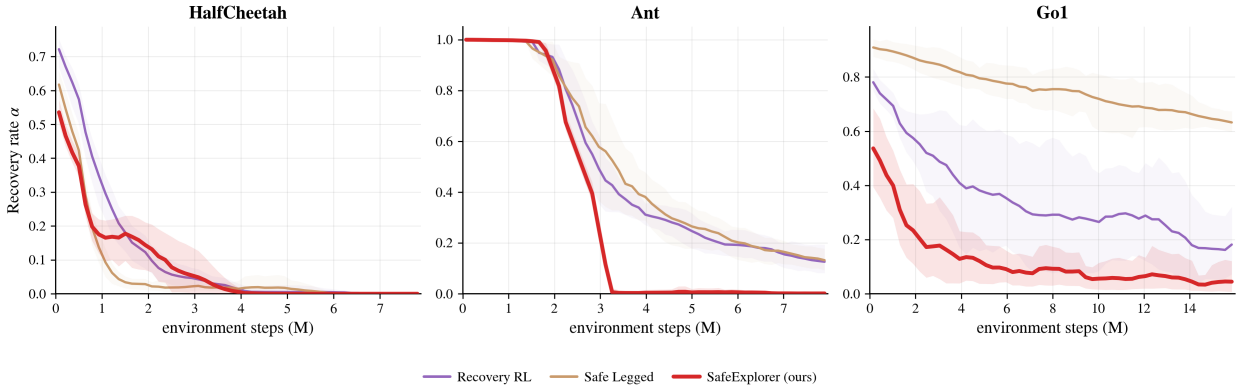

Figure 6: Recovery rate α (fraction of rollout steps on which recovery fired) over training, per environment, for the recovery-using methods (the CMDP solvers act with a single policy and never invoke recovery, so they have no α). Solid = mean across seeds, shaded = [min, max] envelope.

E.13 Masked gradient versus IS under a stochastic recovery

With the objective-gap bound checked, we turn to the gradient estimator: Theorem 1 makes the masked estimator exactly unbiased at recovery steps, while per-step IS, where it applies at all, is biased in general (Remark 1); what remains empirical is how the two compare in practice. Theorem 1 holds for any θ-independent recovery; for a deterministic one, not even IS applies (Section 4.1). Here we ask what happens when the recovery *is* stochastic, so IS becomes an option. Writing $w_t = \pi_\theta(a_t \mid s_t)/\pi^{\mathrm{rec}}(a_t \mid s_t)$ for the importance weight at a recovery step, the IS estimator of $\nabla_\theta J^{\mathrm{mix}}(\theta)$ (Corollary 2) decomposes as

$$\hat{g}_{\mathrm{IS}} = \underbrace{\sum_{t:\, s_t \in \mathcal{R}} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A_t}_{\hat{g}_{\mathrm{masked}}} + \sum_{t:\, s_t \notin \mathcal{R}} w_t\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A_t, \tag{15}$$

with A_t the PPO advantage estimate. The second sum would be mean zero by the score-function identity only if A_t were replaced by an action-independent baseline $b(s_t)$; with the actual advantage, which depends on a_t through the reward and the successor state, its conditional mean is the off-policy policy-gradient contribution at the unsafe states, which is generically nonzero. The per-step IS estimator is therefore biased for $\nabla_\theta J^{\mathrm{mix}}$, and masking removes a bias-carrying term, not merely a noisy one. A recovery policy differs sharply from the main policy by design, so the term's weights are heavy-tailed; truncating them (Espeholt et al., 2018; Munos et al., 2016) trades variance for further bias. The measurement below therefore compares two estimators that do not share a mean: the recorded explosion of the IS weights is evidence about the conditioning of the added term, not a like-for-like variance comparison between unbiased estimators.

Concretely, we rerun SafeExplorer on all three environments (settings of Section 6) with the recovery action *sampled* from the SAC recovery rather than taken greedily at its mean, and shadow-compute, per rollout, the weights' effective sample size and each estimator's gradient variance. Training itself always uses the masked gradient.

Figure 7 shows that the three regimes expose complementary failure modes. On HalfCheetah the strong recovery makes the weights degenerate from the first update. The effective sample size never exceeds 0.5% of the recovery steps, and the IS gradient variance exceeds the masked variance by a factor of up to 8×10^{12}. On Ant the weak recovery initially overlaps the main policy and IS stays formally usable for longer, but the variance ratio holds near 10^{13} for the first 1.2M steps. Go1 completes the spectrum. Over its 12-dimensional action space the recovery's actions are so unlikely under the main policy that the weights die at once (effective sample size always below 3%, reaching 7×10^{-5}), so the added term vanishes before its variance can build, and the ratio never rises above ~ 40 and sits at one for the full 16M steps, over which the quadruped's recovery rate never reaches zero (Appendix E.12). Once the main policy separates from the recovery the weights vanish and the ratio returns to one. IS proves either catastrophically noisy or identical to the masked estimator, never better in any regime we measure.

Training with the truncated-IS gradient (w_t clipped at $c = 10$) confirms the cost (Figure 8). These runs use the sampled-recovery setup of this appendix, so their absolute rewards are not comparable to Table 1's greedy-recovery numbers; only the within-setup masked-vs-IS differences are meaningful. On HalfCheetah the weights die immediately and the two variants are statistically indistinguishable (reward 4675 ± 642 versus 4421 ± 647). On Ant the million-step degeneracy window is expensive: 793 ± 214 reward against 1325 ± 289 for the masked gradient, a 40% drop, and its lower fall count (2066 ± 1582 versus 4682 ± 1047) reflects the weaker policy rather than safer behavior. On Go1 the truncated-IS variant again trails the masked gradient (4884 ± 241 versus 5243 ± 219 reward, a 7% drop) at statistically indistinguishable falls (3293 ± 589 versus 3401 ± 351).

E.14 CMDP baselines: constraint feasibility

Having checked the method's own theory, we finally audit the baselines it is compared against, asking whether the CMDP solvers even satisfy the constraint they are given. This appendix supports the CMDP comparison of Section 7.3. For the comparison to be fair, both solvers use the OmniSafe implementation (Ji et al., 2024) and run on the identical environments, reward functions, network architecture, optimizer, and

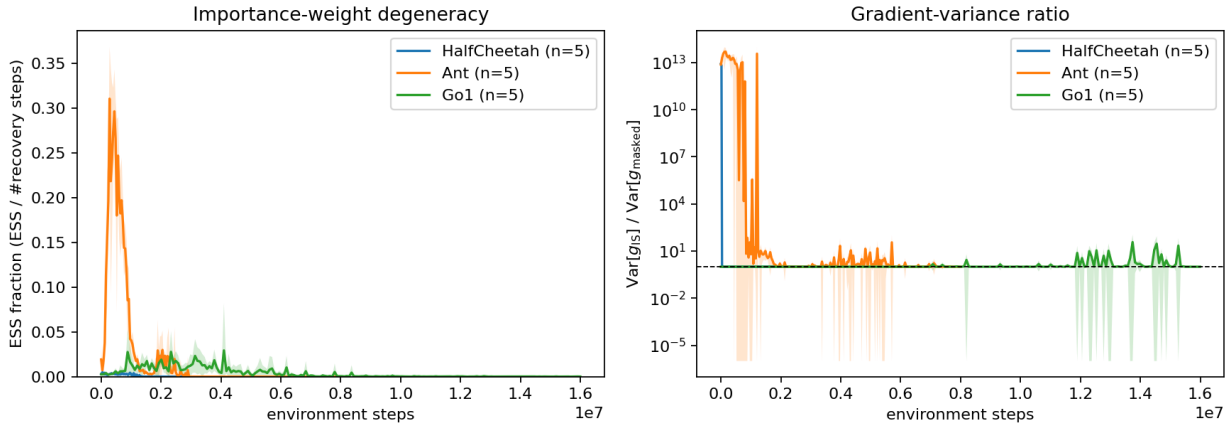

Figure 7: Masked gradient versus IS on a stochastic recovery, SafeExplorer (mean ± std). *Left*: the effective sample size of the importance weights collapses toward zero when the recovery is active, the signature of weight degeneracy. *Right*: the ratio of IS to masked gradient variance (log scale; dashed line at one). The masked estimator targets $\nabla_\theta J^{\text{mix}}$ (Equation (15)) without the IS term's bias and variance pathologies. The HalfCheetah and Ant curves end where the recovery stops engaging, because the diagnostics exist only on updates with recovery steps; on Go1 recovery stays mildly engaged throughout, so the curve spans the full run.

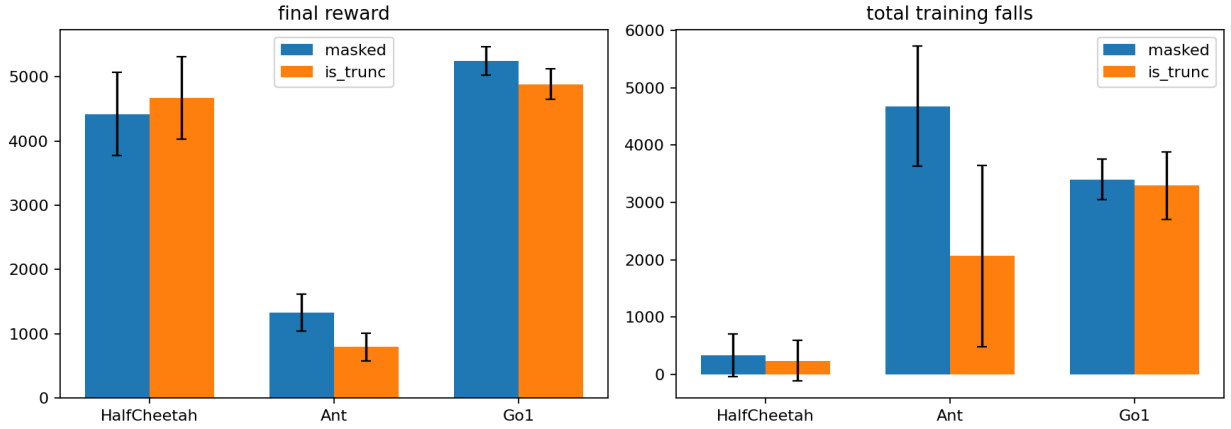

Figure 8: Training SafeExplorer with the masked gradient versus the truncated-IS gradient (mean ± std). *Left*: final reward. *Right*: total training falls.

step budget as the variants in Table 1. We use the direct cost encoding, a cost of 1 on a fall (a termination), the actual safety signal the constraint should bound. The only incidental differences from SafeExplorer are observation and reward normalization, applied to the CMDP baselines and not to SafeExplorer, so they favor the baselines. Reported rewards for both solvers are the raw episode return (`EpRet` in OmniSafe), un-normalized and therefore on the same task-reward scale as the other methods. The normalization affects only the solvers' internal optimization, not the reported numbers.

On this matched footing, Table 16 reports the final episodic cost each solver converges to, against the cost budget of 0.05 it was given (a per-episode fall rate), averaged over the same seeds. The pattern is clean. Both solvers reach feasibility on HalfCheetah, where staying safe is easy, and both *fail* to reach feasibility on Ant and Go1, where the converged cost exceeds the budget by up to 19×. The poor safety of the CMDP baselines is therefore not a tuning artifact. The constrained optimization does not find a feasible policy at all in the regimes where the safe-region intervention is needed.

Table 16: Converged episodic cost vs. the cost budget (≤ 0.05) for the CMDP baselines (final 5% of training). ✓ feasible, × budget violated.

Method	HalfCheetah	Ant	Go1
CPO	0.007 ± 0.011 ✓	0.103 ± 0.029 ×	0.956 ± 0.025 ×
PPO-Lagrangian	0.001 ± 0.001 ✓	0.326 ± 0.177 ×	0.377 ± 0.126 ×

A safe-region indicator cost (cost 1 at every step outside $\mathcal{R}(d_{\max})$) is a second natural encoding, but it is comparable to the recovery-using methods only when the recovery policy is active, since that is the setting in which an out-of-region penalty reduces to recovery-active reward shaping. We therefore report the direct termination encoding in the main comparison and leave the recovery-active safe-region variant to future work.

F Broader Impact

The goal of this work is to make reinforcement learning on physical robots safer to train. Achieving that goal would reduce hardware damage during training, lower the experimental cost borne by robot-learning research groups, and provide a foundation for on-robot continual learning that does not depend on extensive pre-training in simulation.

Set against that intent, the dual-use surface of the technique is narrow. It is dual-use only in the trivial sense that any improvement in robotic control technology is so, because the method does not change the capabilities a deployed policy can express, only the training-time safety profile of how that policy is reached. For this reason we do not foresee specific misuse pathways introduced by this work that are not already present in the underlying PPO + SAC + MuJoCo / Go1 stack.

The experiments themselves are deliberately scoped to limit any training-time harm. We do not collect or release human-subject data and do not perform experiments that could affect bystanders during training. All experiments are in simulation (MuJoCo and the Go1 model), so the training-time falls we reduce are simulated rather than physical; carrying that reduction onto hardware is future work (Section 9).

Because the contribution is methodological, its value depends on being reproducible and affordable, and we account for both: the full algorithm, hyperparameter, and environment specifications needed to rerun the experiments are given in Appendices B to D. We report negative results alongside the positive ones, the soft-gate variants, the fixed-d concessions, and the action-noise stress failures (Appendices E.4, E.6 and E.11), so that future work can avoid repeating them. The compute budget for the component ablation (9 variants $\times$ 3 environments $\times$ 5 seeds, 4 cores each, weighted by per-environment wall-clock) is approximately 3,700 CPU-core-hours, modest relative to contemporary safe-RL benchmarks; the additional sensitivity sweeps in Appendix E add further compute.