CAPSULE: Control-Theoretic Action Perturbations for Safe Uncertainty-Aware Reinforcement Learning

Anonymous Author(s)
Submission Id: «submission id»

ABSTRACT

Ensuring safe exploration in high-dimensional systems with unknown dynamics remains a significant challenge. Existing safe reinforcement learning methods often provide safety guarantees only in expectation, which can still lead to safety violations. Control-theoretic approaches, in contrast, offer hard constraint–based safety guarantees but typically assume access to known system dynamics or require accurate estimation of control-affine models. In this paper, we propose a safe reinforcement learning framework that learns a probabilistic control-affine dynamics model in an offline setting. The learned model is leveraged to explicitly construct control barrier functions (CBFs) that incorporate model uncertainty to provide conservative safety constraints. These CBF constraints are enforced through an online constraint-based action correction mechanism, enabling safe exploration without overly restricting task performance. Empirical evaluations on nonlinear, complex continuous-control benchmarks demonstrate that our approach achieves returns comparable to those of existing baselines while significantly reducing safety violations.

KEYWORDS

Safe Reinforcement Learning, Control Barrier Functions, Control Affine, Offline Pretraining, Probabilistic Ensemble

ACM Reference Format:
Anonymous Author(s). 2026. CAPSULE: Control-Theoretic Action Perturbations for Safe Uncertainty-Aware Reinforcement Learning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 7 pages.

1 INTRODUCTION

Reinforcement Learning (RL) studies the problem of sequential decision-making through interactions between an agent and its environment [13]. At each time step, the agent observes the current state, selects an action, and influences the subsequent evolution of the environment. In return, the environment provides feedback in the form of a scalar reward. Unlike supervised learning, RL does not rely on predefined labels; instead, the agent improves its behavior through trial-and-error interactions. A defining characteristic of RL is the presence of temporal dependencies, where actions affect not only immediate rewards but also future states and long-term outcomes. The objective of an RL agent is to learn a policy that maximizes the expected cumulative reward over time. RL methods have demonstrated remarkable success across a wide range of challenging domains [6, 10, 11].

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). This work is licenced under the Creative Commons Attribution 4.0 International (CC-BY 4.0) licence.

RL agents are driven by the objective of maximizing cumulative reward, which can lead to unsafe exploratory behaviors that incur high costs, damage the environment, or harm the agent itself. Without explicit safety considerations, this reward-driven optimization often encourages high-risk actions that push the agent into hazardous states and cause safety violations. This becomes especially critical in domains like autonomous driving, where unsafe exploration or policy deployment can endanger human lives. For example, a self-driving car must avoid collisions and respect speed limits rather than adopting aggressive maneuvers that cause accidents. Safe Reinforcement Learning (Safe RL) addresses these limitations by introducing mechanisms that enforce safety constraints during training, thereby keeping exploration within acceptable risk boundaries while still enabling effective policy learning. By embedding these constraints directly into the learning process, Safe RL provides a more reliable foundation for deploying RL systems in safety-critical real-world environments.

Exisiting Safe RL methods seeks to maximize reward while enforcing expected safety constraints. However, in highly safety-critical settings, such expectation-based constraints are often insufficient—an agent may still produce unsafe actions as long as they are rare on average. Because such frameworks cannot guarantee strict constraint satisfaction, this motivates looking toward control-theoretic methods within RL, which are specifically designed to provide stronger, more reliable safety guarantees.

In [3, 4], the authors enforce safety by solving a convex optimization that generates a corrective safety-preserving action on top of a base RL policy. Although effective, these approaches generally assume known system dynamics and are typically evaluated in low-dimensional or simplified settings. L1-MBRL [12] reports that enforcing a control-affine structure, where the learned dynamics depend linearly on the action. This can lead to unstable training and degraded performance. These issues become more prevalent in high-dimensional continuous-control environments, where the true dynamics are complex and highly nonlinear. As a consequence, ensuring safety guarantees under unknown dynamics remains a central challenge.

To address these limitations, this paper proposes **C**ontrol-theoretic **A**ction **P**erturbations for **S**afe **U**ncertainty-aware Reinforcement **LE**arning dubbed as **CAPSULE**. This approach to explicitly learn a control-affine dynamics model suitable for online constraint-based safety filtering while avoiding the instability observed in prior online learning approaches. Our key idea is to decouple model learning from policy optimization through an offline pretraining phase that leverages a heteroscedastic probabilistic ensemble of dynamics models. This approach stabilizes the learning of the control-affine model, leveraging offline data before any online interaction begins. Offline pretraining mitigates the model bias reported in earlier works, yielding a much more stable model that is structurally compatible with

our framework. The resulting model provides uncertainty estimates necessary for making safety-critical decisions in subsequent online learning. Unlike [3], which relies on Bayesian models such as the Gaussian Process (GP) Model, i.e, effective but limited to low-data and low-dimensional regimes, our heteroscedastic ensemble provides scalable, PETS-style [5] uncertainty estimates suitable for modern continuous-control benchmarks. This uncertainty enables us to construct state-dependent safety margins and maintain a conservative approach. This allows our framework to enforce safety guarantees even in the presence of model error. Thus, the agent benefits from the exploration and performance while maintaining provable safety constraints derived from the learned dynamics and their uncertainty.

Our contributions are of threefold:

- **Uncertainty Aware Control-Affine Probabilistic Ensemble Dynamics Model:** An offline-trained probabilistic dynamics model that maintains control-affine structure for higher dimensional environments along with heteroscedastic uncertainty estimate for safety guarantees.
- **Online Safe RL via CBFs using learned models:** A hybrid method that pairs Reinforcement Learning with a CBF based safety correction using a pre-trained model allowing for hard-constraint satisfaction which is required in safety critical settings.
- **Evaluation on Complex Continuous-Control Domains:** Experimentation over complex settings such as Safety Gymnasium (MuJoCo based) showing improved performance and fewer safety violations compared to existing safe RL approaches.

2 BACKGROUND

2.1 Markov Decision Process (MDP)

MDP is formulated by the tuple $M = < S, A, R, \mathcal{P}, \gamma >$ where S is the state space, A is the action space and $R : (S \times A) \to [r_{\min}, r_{\max}]$ is the reward function such that $r_{\min}, r_{\max} \in \mathbb{R}$. Further, $\mathcal{P} : (S \times A \times S) \to [0, 1]$ captures the transition dynamics of the environment i.e $\mathcal{P}(\cdot|s, a)$ is a probability distribution over the state space that outputs the probability of agent traversing into next state s' given current state s and action taken a. Discounting parameter is given by $\gamma \in [0, 1)$.

A policy $\pi : (S \times A) \to [0, 1]$ is a function that outputs the probability of taking an action a in the state s i.e a mapping $\pi(\cdot \mid s)$ from states to probability distribution over actions. The objective of a reinforcement learning (RL) agent is to learn a policy π^* that maximizes the expected discounted return given by $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$. The state-value function, or simply the value function, $V^\pi(s)$, under a policy π, is defined as the expected return when starting from state s and following policy π thereafter:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s\right] \tag{1}$$

The Q-value function, or action-value function, $Q^\pi(s, a)$ under a policy π, is defined as the expected return (cumulative discounted reward) starting from state s, taking action a, and thereafter following policy π:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a\right] \tag{2}$$

Trust Region Policy Optimization. TRPO is an on-policy actor-critic algorithm developed to improve the stability and reliability of policy optimization. Standard policy gradients can take large, unconstrained steps in parameter space, causing abrupt changes in the policy. TRPO mitigates this issue by restricting each policy update within a trust region, defined by a Kullback–Leibler (KL) divergence constraint, ensuring that the new policy does not deviate excessively from the previous one and thereby achieving more stable and monotonic improvement. Formally, TRPO maximizes the expected advantage under the old policy, subject to a constraint on the Kullback–Leibler (KL) divergence between successive policies:

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{s,a\sim\pi_{\text{old}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s, a)\right] \\ \text{subject to} \quad & \mathbb{E}_{s\sim\pi_{\text{old}}} \left[D_{\text{KL}}(\pi_{\text{old}}(\cdot|s) \parallel \pi_\theta(\cdot|s))\right] \le \delta \end{aligned} \tag{3}$$

Here, $A^{\pi_{\text{old}}}(s, a)$ denotes the advantage function, and δ controls the maximum policy divergence. By enforcing this KL constraint, TRPO achieves monotonic improvement guarantees and more stable policy updates compared to vanilla policy gradient or standard actor-critic methods.

Constrained Markov Decision Process (CMDP). is an extension of an MDP framework with constraint set C. Where constraint set C is $C = \{c_i, b_i\}_{i=1}^n$ In the context of safe reinforcement learning, the goal is to optimize the value function max $\mathbb{E}[G_t]$ where $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ while also ensuring safety by avoiding violations or risky actions. Let V_M denote the value function of the MDP, M and safe RL objective is defined as shown in the Equation 4, to maximize the V_M

$$\begin{aligned} &\text{maximize: } V_{\mathcal{M}}^{\pi}(\boldsymbol{s}_t) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau r(s_{t+\tau}) \mid \boldsymbol{s}_t\right] \\ &\text{subject to: } \quad h(\boldsymbol{s}_{t+\tau}) \ge b_i, \quad \forall \tau = [0, \infty] \end{aligned} \tag{4}$$

where b is a threshold, and h is a safety function $h : S \to R$, which is not known a prior. $h = J^c(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t c(s_t, a_t)\right]$.

Control Barrier Functions. Consider a nonlinear control-affine system with state $s \in \mathbb{R}^n$ and action $a \in \mathbb{R}^m$, whose dynamics are given by $\dot{s} = f(s) + g(s)a$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz functions.

Safety constraints are specified by a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, which induces the safe set $C : \{s \in \mathbb{R}^n | h(s) > 0\}$. The function h is called a Control Barrier function(CBF) , for all s in a neighbourhood of C, the following condition holds:

$$h(s_{t+1}) \ge (1 - \alpha)\, h(s_t), \quad \alpha \in (0, 1] \tag{5}$$

$$h(f(s_t) + g(s_t)a) \ge (1 - \alpha)\, h(s_t) \tag{6}$$

This inequality defines a state-dependent constraint on the action a, ensuring that the safe set C is forward invariant. If the system starts within C, any action satisfying the above condition guarantees that the state remains safe for all future time.

In practice, the action that is produced by a policy $a = \pi(s)$ can be modified by projecting it onto the set of admissible actions defined by the CBF constraint, typically via a Quadratic Program

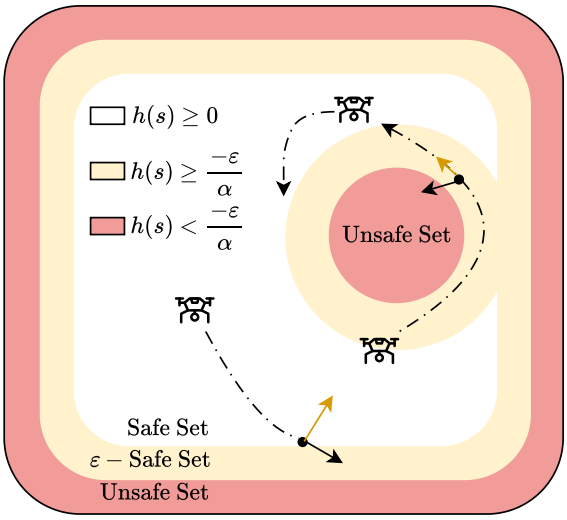


Figure 1: Illustration of the safe, ϵ-safe, and unsafe sets induced by a control barrier function.

(QP). This allows CBFs to be integrated with learning-based or optimal controllers while preserving safety by construction.

Forward Invariance. refers to the property of a set where if the system starts in that set, it stays in that set, it stays in it for all future time under the given dynamics and control input. As we define C is a safe set, the set C is forward invariant under a control law a if: $s(0) \in C \implies x(t) \in C \; \forall t \geq 0$. i.e, the trajectory never leaves the safe set once it starts inside it.

CBFs are explicitly designed to enforce safety constraints over time. To achieve the safety, CBFs enforces forward invariance of the set.

Set Definitions in CBFs: If $h : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function used to define the safe set. As illustrated in Figure 1, the following are the set definitions of CBFs:

- **Safe Set**, $C \triangleq \{x \in \mathbb{R}^n \mid h(x) \geq 0\}$.
- **Interior** of the safe set, $\text{Int}(C) = \{x \mid h(x) > 0\}$.
- **Boundary** of the safe set, $\partial C = \{x \mid h(x) = 0\}$.
- **ϵ-Safe Set**, $C^\epsilon \triangleq \{x \in \mathbb{R}^n \mid h(x) \geq \frac{-\epsilon}{\alpha}\}$.
- **Unsafe set**, $C^u \triangleq \{x \in \mathbb{R}^n \mid h(x) < \frac{-\epsilon}{\alpha}\}$

any control input satisfying the CBF constraint renders the safe set $C \triangleq \{x \mid h(x) \geq 0\}$ forward invariant. Moreover, enforcing the relaxed CBF condition corresponding to the ϵ-safe set $C^\epsilon = \{x \mid h(x) \geq -\epsilon/\alpha\}$ guarantees forward invariance of C^ϵ, ensuring that trajectories initialized within C^ϵ do not enter the unsafe set C^u.

3 PROPOSED METHOD

3.1 Offline Pretraining

Learning the dynamics model online during policy optimization leads to the controller using unstable dynamics and can pose issues like policy divergence [12] and unsafe exploration. To enable safe policy learning without access to explicit system dynamics, we train a dynamics model in an offline manner prior to actual online policy learning. We define the underlying system as a parameterized control-affine probabilistic model:

$$\Delta s_t \mid s_t, a_t \sim \mathcal{N}\big(\mu_\theta(s_t, a_t),\ \sigma_\theta^2(s_t)\big) \tag{7}$$

with control-affine mean:

$$\mu_\theta(s_t, a_t) = \hat{f}_\theta(s_t) + \hat{g}_\theta(s_t) a_t \tag{8}$$

and state-dependent aleatoric uncertainty $\sigma_\theta(s)$, where $\hat{f} : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ models the drift dynamics, and $\hat{g} : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|\times|A|}$ characterizes the effect of action. We avoid $\sigma_\theta(s, a)$ since it complicates learning and yields poor uncertainty estimates from limited samples. The choice of predicting the change in state rather than the next state is to stabilize training since the model learns to predict only the local drift dynamics, which are typically translation-invariant and do not depend on the absolute placement of the state in space. For learning, we utilize the loss function

$$\mathcal{L}(\theta) = -\sum_{n=1}^{N} \log \mathcal{N}\big(\Delta s_n \,\big|\, \mu_\theta(s_n, a_n), \sigma_\theta(s_n)\big) \tag{9}$$

which then on simplifying becomes

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\|\Delta s_n - \mu_\theta(s_n, a_n)\|^2}{2\sigma_\theta(s_n)^2} + \log\big(\sigma_\theta(s_n)^2\big)\right) \tag{10}$$

We incorporate model ensembling to account for epistemic uncertainty in the dynamics. Note that this doesn't affect the control-affine structure that we previously defined. The final formulation thus comes out as given:

$$\bar{\mu}(s_t, a_t) = \bar{f}(s_t) + \bar{g}(s_t)\, a_t, \bar{\sigma}(s_t) \tag{11}$$

where

$$\bar{f}(s_t) = \frac{1}{K}\sum_{k=1}^{K} \hat{f}_{\theta_k}(s_t), \qquad \bar{g}(s_t) = \frac{1}{K}\sum_{k=1}^{K} \hat{g}_{\theta_k}(s_t),$$

and the ensemble-averaged aleatoric uncertainty is

$$\bar{\sigma}(s_t) = \frac{1}{K}\sum_{k=1}^{K} \sigma_{\theta_k}(s_t).$$

This formulation helps us to capture total uncertainty in the learned model [5] which can then be utilized for uncertainty-aware safe exploration. To learn these models we use an offline dataset D_{offline} composed of transitions of form (s_t, a_t, s_{t+1}) collected from rollouts or demonstrations. The dataset is supposed to provide sufficient state coverage in order for the model to generalize better. The pretrained dynamics model is later used to construct the safety barrier used alongside online reinforcement learning.

3.2 Safety Barrier Construction

We employ CBFs to enforce safety during online learning. These encode the safety constraints that must be satisfied while the RL agent is exploring the environment. The resolution of CBFs is relatively less computationally expensive and is solved using QP solvers as demonstrated in [2–4]. These are compatible with the control-affine dynamics model and can be used to solve for safety-corrective action update.

Computation of the CBF requires knowledge of the dynamics model

Algorithm 1 Offline Pretraining

Input: Offline dataset $\mathcal{D}_{\text{offline}}$, ensemble size K, pretraining steps P
Initialize ensemble parameters $\{\theta_1, \theta_2, \ldots, \theta_K\}$ for $\{\hat{f}_{\theta_k}, \hat{g}_{\theta_k}, \sigma_{\theta_k}\}$
for $i = 1, 2, \ldots, P$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    Sample mini-batch $\{(s_n, a_n, s_{n+1})\} \in D_{\text{offline}}$
    Compute $\Delta s_n = s_{n+1} - s_n$
    Update θ_k using $\nabla_{\theta_k} \mathcal{L}(\theta_k)$ in (10)
  end for
end for
Compute ensemble averages $\bar{f}(\cdot), \bar{g}(\cdot), \bar{\sigma}(\cdot)$
Output: Trained ensemble dynamics model $\{\theta_1, \ldots, \theta_K\}$ and ensemble mean functions $(\bar{f}, \bar{g}, \bar{\sigma})$.

to compute the next state and, subsequently, the corrective action. To encounter this, we employ our previously trained ensemble model and use it to predict the next state. Directly using the mean prediction can lead to over-estimation of safety, as it does not account for the uncertainty in model prediction. To mitigate this issue, we also consider the heteroscedastic uncertainty estimate provided by our probabilistic model. Leveraging the Gaussian modelling assumption of the Δs_t, we take a lower bound on the model error and ensure a conservative safety guarantee.

$$s_{t+1} \mid s_t, a_t \sim \mathcal{N}\big(\mu_\theta(s_t, a_t) + s_t,\ \sigma_\theta^2(s_t)\big)$$

$$\implies s_{t+1} \in \mu_\theta(s_t, a_t) + s_t \pm p_\delta \sigma_\theta(s_t) \tag{12}$$

Where p_δ defines the confidence interval on s_{t+1} and ensures s_{t+1} remains in the required range with a probability of $(1-\delta)$. This is used in the CBF formulation in the following way

$$\sup_{a_t \in A} \Big[h\big(s_t + \bar{f}(s_t) + \bar{g}(s_t) a_t - p_\delta |\bar{\sigma}(s_t)|\big) - h(s_t) \Big] \geq -\alpha\, h(s_t) \tag{13}$$

Also, exact feasibility may not always be possible, for example, if there is no safe next state possible for the agent, we allow a non-negative slack ϵ to permit controlled constraint relaxation to still be in an ϵ-safe set. This provides a graceful degradation of the safety constraints. At each step, the safe control action is obtained by solving

$$a^* = \arg\min_{a,\epsilon}\ a^T a + k\epsilon^2$$

$$\text{s.t.}\ \ h\big(s_t + \bar{f}(s_t) + \bar{g}(s_t) a - p_\delta |\bar{\sigma}(s_t)|\big) - h(s_t) \geq -\alpha h(s_t) - \epsilon, \quad \epsilon \geq 0$$

where $k >> 0$ penalizes slack usage and regulates how strictly the barrier constraint is enforced. When there is a valid action that satisfies the constraint, the optimal solution has $\epsilon = 0$. This formulation allows the learned CBF constraint to be enforced consistently during online RL Policy training while accommodating model approximation error and control limitations. We leverage the pre-trained dynamics model along with the safety barrier constraints in online policy optimization.

Algorithm 2 CAPSULE

Input: pretrained ensemble models $\bar{f}, \bar{g}, \bar{\sigma}$
Initialize: policy buffer B, policy network π_θ^{RL} , Compensator $\pi_{\phi_k}^{bar}$, Compensator buffer $\hat{\text{B}}$
for $k = 1, 2, 3$....Epochs **do**
  Initialize state $s_0 \sim \rho$
  for $t = 1, \ldots, T$ **do**
    Sample $a_t^{\text{RL}} \sim \pi_k^{\text{RL}}(s_t)$
    Sample $a_t^{\text{bar}} \sim \pi_k^{\text{bar}}(s_t)$
    Solve for $a_t^{\text{CBF}}(s_t, a_t^{\text{RL}} + a_t^{\text{bar}})$ using (15)
    Execute a_t from (14) to get (s_{t+1}, r_t)
    Store $(s_t, a_t^{\text{CBF}} + a_t^{bar})$ in $\hat{\text{B}}$
    Store (s_t, a_t, s_{t+1}, r_t) in B
  end for
  Train RL policy π_k^{RL} using B to get θ_{k+1}
  Train Compensator π_k^{bar} using $\hat{\text{B}}$ to get ϕ_{k+1}
end for
return $\pi_\theta^{\text{RL}}, \pi_\phi^{bar}$

3.3 Online Policy Optimization

In this section, we outline the online training phase, where the agent interacts with the environment. During this stage, the objective is to optimize the task reward while ensuring that all the executed actions satisfy the safety constraints. Rather than executing the raw actions produced by the RL policy, the system employs a corrective control structure that modifies actions when necessary to maintain safety.

After each training epoch, the policy network π_θ^{RL} is updated using a standard RL algorithm such as SAC, TRPO, or PPO. In this work, we employ TRPO as the base RL algorithm, yielding an updated task-driven controller $\pi_{\theta_{k+1}}^{\text{RL}}$.

However, executing $\pi_{\theta_k}^{\text{RL}}$ directly will lead to constraint violations. To mitigate this, our method incorporates two additional corrective components: (1) a safety compensation term a_t^{bar} inspired from [3], that incrementally approximates the accumulated history of past CBF corrections, and (2) a real-time CBF controller a_t^{CBF}, computed by solving a constrained quadratic program using the pre-trained dynamics model. At each timestep t, the system first samples a candidate action a_t^{RL} from the current policy. The compensator update a_t^{bar} that accounts for safety history is added as correction to a_t^{RL}. Eventually, the Compensator estimates the safety provided by the CBF corrections and reduces reliance on CBF Controller. Finally, the CBF controller minimally adjusts this action to enforce forward invariance of the safe set, or if not possible, then in the ϵ-safe set. The resulting executed action is the composition:

$$a_t^{\text{safe}} = a_t^{\text{RL}} + a_t^{bar} + a_t^{\text{CBF}}(s_t,\ a_t^{\text{RL}} + a_t^{bar}) \tag{14}$$

The resulting optimization equation to solve the CBF then becomes

$$\begin{aligned} a_t^{\text{CBF}} &= \arg\min_{a,\epsilon}\ a^T a + k\epsilon^2 \\ \text{s.t.}\ \ & h\Big(s_t + \bar{f}(s_t) + \bar{g}(s_t)(a_t^{\text{RL}} + a_t^{bar} + a) - p_\delta |\bar{\sigma}(s_t)|\Big) \\ & - h(s_t) \geq -\alpha h(s_t) - \epsilon, \quad \epsilon \geq 0 \end{aligned} \tag{15}$$

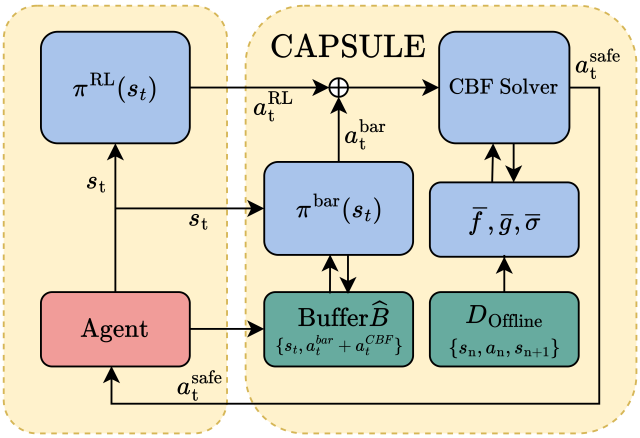


Figure 2: The proposed CAPSULE Algorithmic flow

All the executed transitions are stored in a replay buffer B to update the RL policy, whereas the tuples $(s_t,\ a_t^{\mathrm{CBF}} + a_t^{bar})$ are stored separately in $\hat{B}$ to refine the compensator. The complete online training loop is summarized in Algorithm 2.

4 EXPERIMENTS

Experiments were conducted in continuous-control environments from MuJoCo which are illustrated in Figure 3. To obtain a reliable dynamics approximation prior to online learning, we first perform offline pre-training using a dataset of 1M transitions. The corresponding loss curves and learned dynamics accuracy are presented later in the Results section. Comparisons with fully non-linear ensemble model have been given to show we are able to learn good control-affine representations.

After pre-training, the agent is trained online using the proposed CBF-augmented reinforcement learning framework. For empirical evaluation, we compare our method against state-of-the-art Safe Policy Optimization algorithms. We intentionally focus on these CMDP-based baselines because other Safe RL approaches with hard safety constraints typically rely on requirements like known system dynamics, a pre-computed backup (safe) policy, or human interventions during training, and have shown limited success in complex higher-dimensional environments.

We implement these baselines with their safety thresholds modified to ≤ 0, [14], leading to stricter safety conditions thus leading to a fair comparison between the algorithms.

4.1 Experimental Setup

We evaluate our approach under the *SafeVelocity* benchmark introduced in *Safety Gymnasium* [7]. In this setting, agents operate in MuJoCo environments and are required to balance task performance with consideration to velocity limits. As described in [7], these tasks encourage agents to move faster while staying alive (not falling, being in range etc.) to obtain higher rewards while simultaneously enforcing safety by assigning a cost whenever a velocity threshold is exceeded.

4.2 Baselines

To assess the effectiveness of our method, we benchmark against established safe RL baselines. For fair comparison, all methods use identical reward functions, safety specifications, and environment configurations.

- **TRPO-Lag** [9]: A constrained variant of TRPO that incorporates a Lagrangian dual variable to enforce expected cost constraints while performing trust-region policy updates.
- **FOCOPS** [18]: A first-order constrained optimization method that stabilizes safe policy learning via a KL-regularized objective and a cost-sensitive update rule.
- **PPO-Lag** [9]: A Lag adaptation of PPO that uses the same dual-variable mechanism as TRPO-Lagrangian but applies it within PPO's clipped policy gradient framework, making it more computationally efficient.

4.3 Results

As shown in Figure 4, the offline pretraining results across all benchmark environments indicate that our control-affine ensemble model approximates the system dynamics with performance comparable to that of a general nonlinear ensemble model. All the loss curves during training are computed using the loss function defined in (10).

We trained CAPSULE and all baseline algorithms for 10^6 environment steps, and all reported metrics are averaged over five random seeds, with variance included to ensure statistical reliability. As shown in Figure 5, across all environments our method substantially reduces the number of safety violations.

In the Hopper environment (Figure 5(a)), CAPSULE achieves the fewest violations among all baselines, whereas TRPO-Lag exhibits the highest number of violations. As seen in Figure 5(d), TRPO-Lag also accumulates higher rewards precisely because it does not adhere to safety constraints, leading to safety violations.

A similar trend appears in the Walker environment (Figure 5(b) and 5(e)), where unsafe behavior often results in early termination. CAPSULE maintains competitive task performance while drastically lowering violations, capturing the inherent reward–safety trade-off in these environments. TRPO-Lag again demonstrates higher returns at the cost of a significantly increased violation count.

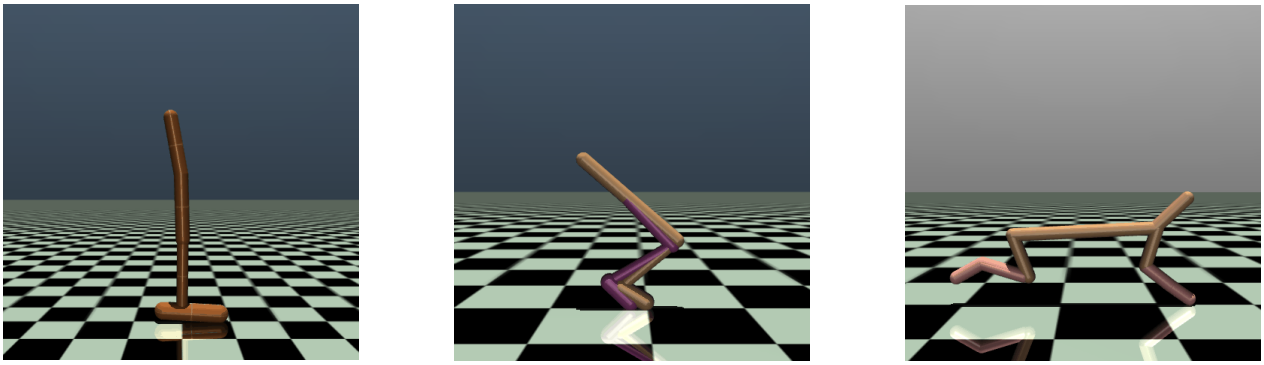
In the HalfCheetah environment, the episode does not terminate when the agent topples or exceeds velocity limits. As a consequence, as shown in Figure 5(c) and 5(f), agents can accumulate high rewards while simultaneously incurring a large number of violations. This property explains the substantial increase in violations across algorithms. Even in this setting, CAPSULE incurs fewer violations and maintains performance comparable to the baselines.

All presented results are based on evaluation trajectories, with 10 evaluation episodes generated after each training update.

5 RELATED WORK

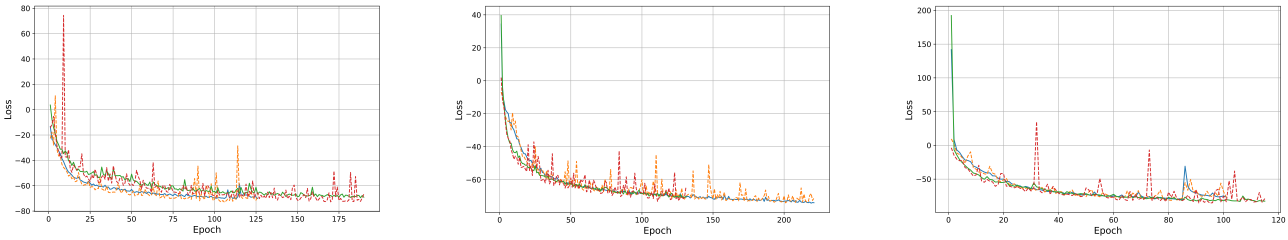
5.1 Safe Policy Optimization

The authors in CPO[1] proposed the first policy-gradient methods to address CMDPs in deep reinforcement learning setting. CPO



(a) Hopper (b) Walker (c) HalfCheetah

Figure 3: Visualizations of different MuJoCo control environments used in our experiments: Hopper, Walker, HalfCheetah.



(a) Hopper (b) Walker (c) Halfcheetah

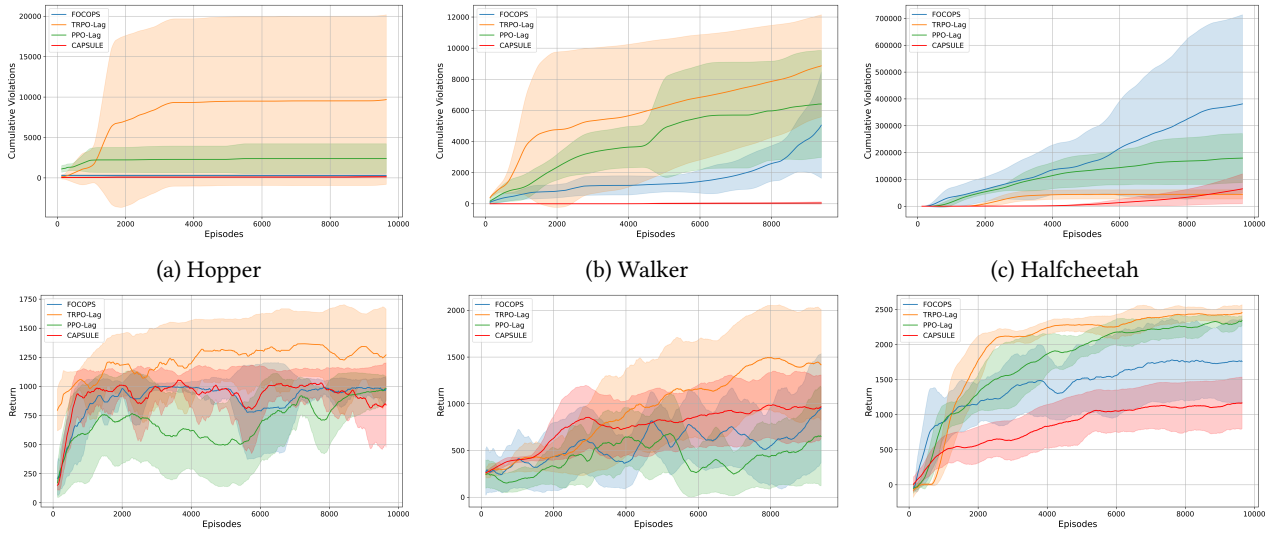
CAPSULE train loss
CAPSULE val loss
Non linear model train loss
Non linear model val loss

Figure 4: Pre-training results on MuJoCo continuous control environments.

enforces safety constraints while guaranteeing monotonic improvement in expected reward. However, CPO incurs substantial computational overhead due to the need to estimate the Fisher information matrix and optimize a second-order approximation of the objective. Additionally, approximation bias and sampling noise can adversely affect performance. PCPO [15] adopts a two-step strategy that first improves the policy using TRPO [10] and subsequently projects the updated policy onto a constraint-satisfying set to ensure safety. This approach demonstrates improved empirical performance over CPO in certain settings. Nevertheless, PCPO relies on second-order optimization in both stages, resulting in a higher computational cost. On the other hand, FOCOPS [18] apply a primal-dual approach to CMDPs by performing policy optimization in a non-parametric space, followed by projection back into the parameterized policy class. While this method is comparatively simple to implement and often exhibits improved sample efficiency, it remains susceptible to instability. All the above method formulations are based on safety in expectation, and may encounter safety violations. To address this issue, we have proposed a control-theoretic approach for safe RL.

5.2 Control-theoretic based approaches

Control Barrier Functions (CBFs) have also been applied to safe reinforcement learning in recent years. Several works aim to reduce or eliminate reliance on known system dynamics by learning safety constraints directly from data. In [16], learns both the policy and the associated CBFs in a data-driven manner, enabling safe policy optimization without access to an explicit environment model. These approaches [8, 17] combine reinforcement learning with CBF-based constraints to ensure safety during learning. [3] use CBFs to explicitly restrict the policy search space, thereby improving both safety and sample efficiency, often supported by Gaussian process–based uncertainty estimates. Other works leverage learned or partially known dynamics together with CBFs to provide formal safety guarantees, accompanied by convergence and stability analyses. Empirical results in these studies demonstrate improved performance and stronger safety guarantees compared to baseline safe RL methods. However, all the above methods have been evaluated on simplistic settings and/or assumed to know the system dynamics a priori. We utilize grounded concepts from CBF along with uncertainty-based stochastic models to overcome these issues and evaluate in challenging environments.



(a) Hopper (b) Walker (c) Halfcheetah

(d) Hopper (e) Walker (f) Halfcheetah

Figure 5: Policy Evaluation on *SafeVelocity* in Mujoco continuos control environments.

6 CONCLUSION

In this work, we introduced CAPSULE, a safe RL framework that employs control-theoretic formulations to provide safety guarantees and leverages uncertainty-aware dynamics modeling to provide reliable predictions. By learning a control-affine ensemble model offline, we obtain a stable approximation of the control-affine dynamics of the system without requiring access to the true dynamics. This allows the framework to satisfy forward invariance during online interaction. More broadly, CAPSULE illustrates how control-theoretic structures and data-driven uncertainty estimation can be combined to support safe exploration in complex environments.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31.

[2] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Automat. Control* 62, 8 (2016), 3861–3876.

[3] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3387–3395.

[4] Yikun Cheng, Pan Zhao, and Naira Hovakimyan. 2023. Safe and efficient reinforcement learning using disturbance-observer-based control barrier functions. In *Learning for Dynamics and Control Conference*. PMLR, 104–115.

[5] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems* 31 (2018).

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

[7] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. 2023. Safety gymnasium: A unified safe reinforcement learning benchmark. *Advances in Neural Information Processing Systems* 36 (2023), 18964–18993.

[8] Zahra Marvi and Bahare Kiumarsi. 2020. Safe off-policy reinforcement learning using barrier functions. In *2020 American Control Conference (ACC)*. IEEE, 2176–2181.

[9] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708* 7, 1 (2019), 2.

[10] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

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

[12] Minjun Sung, Sambhu Harimanas Karumanchi, Aditya Gahlawat, and Naira Hovakimyan. 2024. Robust Model Based Reinforcement Learning Using L_1 Adaptive Control. In *The Twelfth International Conference on Learning Representations*.

[13] Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[14] Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. 2023. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*. PMLR, 36593–36604.

[15] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. 2020. Projection-Based Constrained Policy Optimization. In *International Conference on Learning Representations*.

[16] Yujie Yang, Yuxuan Jiang, Yichen Liu, Jianyu Chen, and Shengbo Eben Li. 2023. Model-free safe reinforcement learning through neural barrier certificate. *IEEE Robotics and Automation Letters* 8, 3 (2023), 1295–1302.

[17] Baohe Zhang, Yuan Zhang, Lilli Frison, Thomas Brox, and Joschka Bödecker. 2024. Constrained reinforcement learning with smoothed log barrier function. *arXiv preprint arXiv:2403.14508* (2024).

[18] Yiming Zhang, Quan Vuong, and Keith Ross. 2020. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems* 33 (2020), 15338–15349.