MODEL-BASED PROACTIVE COST GENERATION FOR LEARNING SAFE POLICIES OFFLINE WITH LIMITED VIOLATION DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning constraint-satisfying policies from offline data without risky online interaction is crucial for safety-critical decision making. Conventional methods typically learn cost value functions from abundant unsafe samples to define safety boundaries and penalize violations. However, in high-stakes scenarios, risky trial-and-error is infeasible, yielding datasets with few or no unsafe samples. Under this limitation, existing approaches often treat all data as uniformly safe, overlooking safe-but-infeasible states-states that currently satisfy constraints but inevitably violate them within a few steps-leading to deployment failures. Drawing inspiration from the concept of knowledge-data integration, we leverage large language models (LLMs) to incorporate natural language knowledge into the policy to address this challenge. Specifically, we propose PROCO, a model-based offline safe reinforcement learning (RL) framework tailored to datasets largely free of violations. PROCO first learns a dynamics model from offline data and constructs a conservative cost function by grounding natural-language knowledge of unsafe states in LLMs, enabling risk estimation even without observed violations. Using the cost function and learned model, PROCO performs model-based rollouts to synthesize diverse counterfactual unsafe samples, supporting reliable feasibility identification and feasibility-guided policy learning. Across a range of Safety-Gymnasium tasks with exclusively safe or minimally risky training data, PROCO integrates seamlessly with a variety of offline safe RL algorithms and consistently demonstrates reduced constraint violations and improved safety performance compared to both the original methods and other behavior cloning baselines.

1 INTRODUCTION

Safe reinforcement learning (RL), which aims to derive policies that satisfy predefined safety constraints, is crucial for real-world applications such as autonomous driving (Zhang et al., 2021), robotic control (Brunke et al., 2022), and aligning large language models (LLM) with human values (Dai et al., 2024). A fundamental limitation of conventional online RL is its reliance on trial-and-error exploration, which is inherently risky and can preclude its deployment in safety-critical settings (Levine et al., 2020). To mitigate these risks, offline safe RL has emerged as a promising paradigm, seeking to learn safe policies exclusively from pre-collected datasets without necessitating hazardous online interactions (Liu et al., 2024; Zheng et al., 2024).

Recently, numerous methods have been proposed for offline safe policy learning. One line focuses on learning conservative cost value functions to penalize unsafe actions (Xu et al., 2022; Guo et al., 2025), while another leverages cost advantage functions to weight behavior cloning (BC) on offline data (Lee et al., 2022; Zheng et al., 2024; Koirala et al., 2025; Chemingui et al., 2025). Both approaches critically rely on abundant unsafe samples for accurate cost estimation and effective policy optimization. However, in many safety-critical high-stakes scenarios, such as autonomous driving or robotics manipulation, collecting large numbers of unsafe samples is impractical, as it is likely to cause detrimental effects on the agent or the environment (Dulac-Arnold et al., 2019), yielding dataset with only few or no unsafe samples. In such cases, existing methods may fail to enforce safety, as they treat all samples equally safe and overlook **safe-yet-infeasible** states. For instance, during robot data collection, to avoid harming the agent, external interventions are used to

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

stop the process whenever the robot approaches a collision with an obstacle, leaving terminal states that are currently safe but would soon violate constraints due to inertia if no external intervention is applied (Bansal et al., 2017). This limitation thus motivates a new problem: **how can we learn a safe policy offline when unsafe samples are scarce or entirely absent?**

The root cause of the aforementioned challenge is, due to the absence of unsafe samples, the dataset lacks a crucial piece of knowledge-namely, which behaviors are unsafe or risky. This observation motivates a knowledge-data integration approach, wherein human or external knowledge is provided to the agent to bridge this gap (Nie et al., 2025). Given that such knowledge typically exists in natural language (Alismail & Lanquillon, 2025), LLMs-with their robust language processing and reasoning capabilities-serve as an ideal vehicle for knowledge processing. Furthermore, to align the LLM-derived knowledge with the empirical data, we incorporate the concept of model rollouts. Specifically, we propose Model-Based **Pro**active **Co**st Generation (PROCO), a model-based offline safe RL algorithm that identifies infeasible states and learns safe policies from datasets containing few or no unsafe samples. PROCO first leverages an LLM to derive a conservative cost function from the natural language knowledge of the constraint, which is then validated and refined using the available safe data and, when present, a limited number of unsafe samples. Subsequently, it learns a dynamics model from the offline dataset. Using the cost function and dynamics model, PROCO simulates future state evolution to generate diverse and informative unsafe samples, enabling efficient detection of infeasible states. Moreover, the conservative cost function enables states in close proximity to unsafe ones to be labeled as unsafe as well, shortening the transition steps from infeasible to unsafe states and thus reducing the influence of model errors on infeasible state identification. Extensive experiments across diverse Safety-Gymnasium environments demonstrate that, when integrated with multiple offline safe RL algorithms, PROCO significantly outperforms both the original algorithms and other behavior cloning-based baselines on safe-only datasets, achieving improvements of over 400% in safety performance.

2 PRELIMINARIES

2.1 SAFE RL AND OFFLINE SAFE RL

In this work, we focus on safe RL under hard-constraints, which can be modeled as a hard-constraint CMDP, defined as a tuple $\langle S, A, r, h, c, P, \gamma \rangle$. Here, S and A denote the state and action spaces, respectively; $r : S \times A \to [-R_{\text{max}}, R_{\text{max}}]$ represents the reward function, $h : S \to [h_{\text{min}}, h_{\text{max}}]$ is the constraint violation function, and $c : S \to \{0, 1\}$ is the cost function. $P : S \times A \times S \to [0, 1]$ specifies the transition dynamics, and $\gamma \in (0, 1)$ is the discount factor. Typically, $c(s) = \mathbb{I}(h(s) > 0)$, which means $h(s) > 0$ is unsafe while $h(s) \leq 0$ is safe. A policy $\pi : S \to \Delta(A)$ maps states to action distributions. Under policy π, the expected discounted reward return and cost return are defined as $R(\pi) = \mathbb{E}_{\tau \sim P_\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ and $C(\pi) = \mathbb{E}_{\tau \sim P_\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t)\right]$, where $\tau = (s_0, a_0, s_1, a_1, \dots) \sim P_\pi$ denotes a trajectory induced by π and the environment dynamics P. Thus, the objective of solving a hard-constraint CMDP is to find a policy that maximizes reward return while ensuring the cost return remains zero.

In the offline setting, we are given an offline dataset $\mathcal{D}$ generated by the behavior policy π_β. Now, the goal is to learning a safe policy purely from this dataset. To avoid extrapolation error in offline RL, the optimization objective is formulated as:

$$\max_{\pi} \; R(\pi), \; s.t. \; C(\pi) \leq 0 \; ; \; D(\pi || \pi_\beta) \leq \epsilon, \tag{1}$$

where $D(\pi||\pi_\beta)$ is a divergence term (e.g., KL divergence $D_{\text{KL}}(\pi||\pi_\beta)$) used to prevent distribution shift. To investigate the problem of learning safe policies under scarce or absent unsafe samples, we focus on scenarios where $\mathcal{D}$ contains no unsafe samples. Optionally, we assume the availability of an extremely small dataset $\mathcal{D}_{\text{unsafe}}$ consisting of no more than 100 unsafe transitions, satisfying $|\mathcal{D}_{\text{unsafe}}| << |\mathcal{D}|$. Meanwhile, in most practical applications, a continuous constraint violation function h is often unavailable. Accordingly, in this work we define h as also a binary function: $h(s) = h_{\text{min}} \leq 0$ if $c(s) = 0$, and $h(s) = h_{\text{max}} > 0$ otherwise. To compensate for the lack of unsafe samples, we assume the availability of a natural language specification of the task's safety constraint, L_{cost}, to provide safety-related information. Thus, the agent must exploit both the safe dataset $\mathcal{D}$ and the constraint description L_{cost} in order to acquire a safe policy.

2.2 HAMILTON-JACOBI REACHABILITY ANALYSIS

To solve hard-constrained safe RL, HJ reachability analysis (Bansal et al., 2017) is one of the widely used approaches. It typically models π as a deterministic policy and defines the feasible set as

Definition 2.1 (Feasible set (Yu et al., 2022))**.** The feasible set of a policy π can be defined as

$$S_f^\pi := \{s \in S | h(s_t^\pi | s_0 = s) \leq 0, \forall t \in \mathbb{N}\}. \tag{2}$$

The largest feasible set S_f^* is a subset of S composed of states from which there exists at least one policy that keeps the system satisfying the constraint, i.e.,

$$S_f^* := \{s \in S | \exists \pi, h(s_t^\pi | s_0 = s) \leq 0, \forall t \in \mathbb{N}\}. \tag{3}$$

Above, $h(s_t^\pi | s_0 = s)$ specifies the state constraint sequence of the trajectory $\{h(s_0^\pi), h(s_1^\pi), \ldots, | s_0 = s, \pi\}$.

Definition 2.2 (Optimal feasible value function (Yu et al., 2022))**.** The optimal feasible state-value function V_h^*, and the optimal feasible action-value function Q_h^* are defined as

$$\begin{aligned} V_h^*(s) &:= \min_\pi V_h^\pi(s) := \min_\pi \max_{t \in \mathbb{N}} h(s_t), s_0 = s, a_t \sim \pi(\cdot | s_t), \\ Q_h^*(s,a) &:= \min_\pi Q_h^\pi(s,a) := \min_\pi \max_{t \in \mathbb{N}} h(s_t), s_0 = s, a_0 = a, a_t \sim \pi(\cdot | s_t), \end{aligned} \tag{4}$$

where V_h^π represents the maximum constraint violations in the trajectory induced by π starting from s. The feasible set and largest feasible set can thus be written as (Zheng et al., 2024)

$$S_f^\pi := \{s | V_h^\pi(s) \leq 0\}, \; S_f^* := \{s | V_h^*(s) \leq 0\}. \tag{5}$$

Based on Definition 2.1, once S_f^* is obtained, the feasibility of states in $\mathcal{D}$ can be determined. Furthermore, Definition 2.2 indicates that S_f^* can be obtained by computing the optimal feasible value function $V_h^*(s)$. Thus, the feasible Bellman operator $\mathcal{B}^*$ (Fisac et al., 2019) is proposed:

$$\mathcal{B}^* Q_h(s,a) := (1-\gamma) h(s) + \gamma \max\{h(s), V_h^*(s')\}, \; V_h^*(s') = \min_{a'} Q_h(s', a'). \tag{6}$$

Although the aforementioned S_f^π and V_h^π are defined for a deterministic policy π, the computation of V_h^* and $\mathcal{B}^*$ is policy-independent and does not rely on whether π is deterministic. Therefore, these quantities can be readily applied to stochastic policies as well (Zheng et al., 2024).

3 METHOD

This section details PROCO, a novel algorithm for offline safe policy learning when unsafe samples are scarce or absent (Figure 1). Section 3.1 describes our dynamics model-based feasibility identification, Section 3.2 presents the motivation and use of LLMs to generate conservative cost functions, and Section 3.3 outlines the overall pipeline for assisting safer policy learning with PROCO. Implementation details can be found in Appendix C.

3.1 FEASIBILITY IDENTIFICATION WITH A DYNAMICS MODEL

Theoretically, an accurate estimate of $V_h^*(s)$ suffices to evaluate the feasibility of state s. However, computing $V_h^*(s)$ via $\mathcal{B}^*$ demands a substantial amount of unsafe samples beyond $\mathcal{D}$. To overcome this limitation, we propose leveraging a learned dynamics model to proactively generate future unsafe samples. First, we train an ensemble dynamics model $\hat{T}$ using the offline dataset $\mathcal{D}$:

$$\min_{\hat{T}} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} [||\hat{T}(s,a) - s'||_2^2]. \tag{7}$$

Subsequently, leveraging $\hat{T}$ together with the dataset $\mathcal{D}$, we perform branched rollout to simulate future trajectories. To address the potential underestimation of V_h^* due to model uncertainty, we introduce a conservative feasible Bellman operator, denoted as $\bar{\mathcal{B}}^*$:

$$\bar{\mathcal{B}}^* Q_h(s,a) := (1-\gamma) h(s) + \gamma \max\{h(s), \max_{s' \in \hat{T}(s,a)} \min_{a'} Q_h(s', a')\}, \tag{8}$$

For $\bar{\mathcal{B}}^*$, we establish the following desirable proposition:

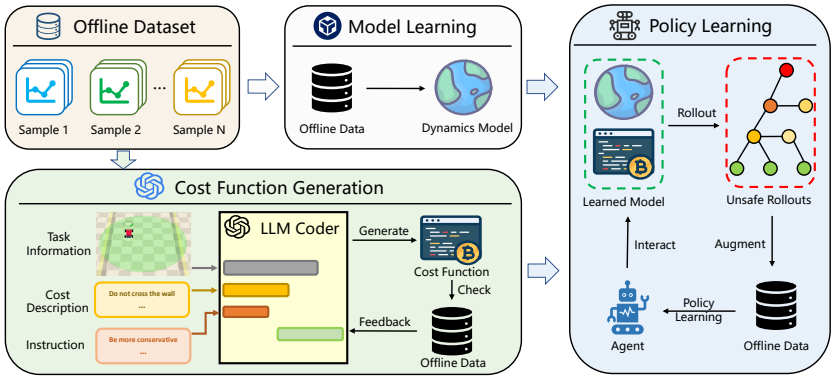


Figure 1: Structure of PROCO.

Proposition 3.1. *$\bar{\mathcal{B}}^*$ is a γ contraction mapping in the $\infty-$norm, and satisfies that $\bar{Q}_h^*(s,a) \geq Q_{h,T}^*(s,a)$ for all (s,a) and for all $T \in \hat{T}$, where $\bar{Q}_h^*(s,a)$ is the convergence result of $\bar{\mathcal{B}}^*$ and $Q_{h,T}^*(s,a)$ is the convergence result of $\mathcal{B}^*$ under transition dynamics T.*

Finally, under certain assumptions, we can guarantee the effectiveness of feasibility identification when employing $\bar{\mathcal{B}}^*$ together with $\hat{T}$:

Assumption 3.2. There exists a horizon $H^* \in \mathbb{N}$ such that, for any infeasible state s, any sequence of actions $a_0, \ldots, a_{H^*-1}$ will lead to an unsafe state.

Assumption 3.3. The learned ensemble dynamics model $\hat{T}(s,a)$ is calibrated, that is, for all $(s,a) \in S \times A$, the ground truth dynamics model $T(s,a)$ satisfies that $T(s,a) \in \hat{T}(s,a)$.

Theorem 3.4. *If Assumption 3.2 and Assumption 3.3 hold, $\forall s \in \bar{S}_f^* = S - S_f^*, \forall a$, the convergence result $\bar{Q}_h^*$ learned by $\bar{\mathcal{B}}^*$ and $\hat{T}$ rollout data satisfies that, $\forall a, \bar{Q}_h^*(s,a) > 0$ for large enough γ.*

3.2 LLM-ASSISTED CONSERVATIVE COST FUNCTION GENERATION

Theorem 3.4 enables feasibility identification for samples in $\mathcal{D}$, yet it critically depends on Assumption 3.3, which requires exact accuracy of the learned dynamics transitions, a condition rarely satisfied in practice. We therefore relax it as Assumption 3.5 and introduce another assumption:

Assumption 3.5. For a given policy π, there exists δ, $\forall s, a$, $\max_t \mathbb{E}_{s\sim p_1^t(s)} D_{\mathrm{KL}}(p_1(s'|s)||p_2(s'|s)) \leq \delta$, where $p_1(s'|s) = T(s'|s, \pi(\cdot|s))$ for $t > 0$ and $p_1(s'|s) = T(s'|s,a)$ for $t = 0$, and $p_2(s'|s) = T'(s'|s, \pi(\cdot|s))$ for $t > 0$ and $p_2(s'|s) = T'(s'|s,a)$ for $t = 0$, T is the ground truth dynamics model, and T' refers to any one of the dynamics model in the ensemble dynamics model set $\hat{T}$.

Assumption 3.6. For a given constraint violation function h, there exists a constant K, such that for any two state distributions μ, v, it satisfies that $|\mathbb{E}_\mu[h(s)] - \mathbb{E}_v[h(s)]| \leq K D_{\mathrm{TV}}(\mu||v)$, where D_{TV} is the total variance distance.

Assumption 3.6 assumes a correlation between the expected constraint violation and the state distribution, such that similar state distributions correspond to similar expected constraint violations. This assumption is generally valid in practice, and based on it, we can proceed to analyze the effectiveness of feasibility identification:

Theorem 3.7. *If Assumption 3.2, Assumption 3.5 and Assumption 3.6 hold, $\forall s \in \bar{S}_f^* = S - S_f^*$, $\forall a$, given π, the convergence result $\bar{Q}_h^*$ learned by $\bar{\mathcal{B}}^*$ and branched rollout data of π in $\hat{T}$ satisfies*

$$\bar{Q}_h^*(s,a) \geq (1-\gamma^{t^*})h_{min} + \gamma^{t^*}\mathbb{E}_{p_1^{t^*}}[h(s)] - \gamma^{t^*} t^* K\delta,\ t^* = \arg\max_{t\in\{1,\ldots,H^*\}} [\mathbb{E}_{p_1^t}[h(s)]]. \quad (9)$$

Remark. Since $\gamma, h_{\min}, K$ are outer given of constant, Theorem 3.7 reveals that the feasible value function of the infeasible state is lower bounded by $t^*, h(s)$ and δ. To ensure safe decision making,

we hope $\bar{Q}_h^*(s,a)$ can be large enough. Since $\mathbb{E}_{p_1^t}[h(s)]$ is upper bounded by $h_{\max}$, a practical way is to lower δ or lower t^*. Here, δ denotes the model rollout error. Due to cumulative error, shorter rollout horizons lead to lower errors. The term t^* represents the number of steps required to transition from an infeasible state s to its associated unsafe state s'. Thus, the distance between s and s' determines t^*, with smaller distances yielding smaller values of t^*. Hence, the analysis indicates that **bringing an infeasible state closer to its corresponding unsafe state reduces the rollout error δ as well as the step horizon t^*, resulting in more efficient feasibility identification**.

Therefore, we propose leveraging an LLM to generate a more conservative cost function, which **marks states near actual unsafe states also as unsafe**, effectively bringing infeasible states closer to unsafe states. Specifically, we first provide three natural language descriptions, L_{task}, L_{cost}, and L_{inst}, to the LLM, enabling it to generate the task's cost function $\bar{c}$:

$$\bar{c} = \text{LLM}(L_{\text{task}}, L_{\text{cost}}, L_{\text{inst}}), \tag{10}$$

where L_{task} is task-related information, such as the meaning of states; L_{cost} is the provided language description of the safety constraint; and L_{inst} provides explicit instructions to the LLM, directing it to generate a cost function that is more conservative than the constraint described in L_{cost}.

Nevertheless, the cost function produced directly by the LLM may not be reliable. To mitigate this, we propose a validation-and-feedback mechanism leveraging both the small unsafe dataset $\mathcal{D}_{\text{unsafe}}$ and the safe dataset $\mathcal{D}$. Concretely, we begin by validating $\bar{c}$ on $\mathcal{D}_{\text{unsafe}}$, requiring 100% accuracy to guarantee that all unsafe samples are correctly identified, thereby eliminating safety risks. After this criterion is satisfied, we evaluate the proportion of safe samples in $\mathcal{D}$ that are classified as unsafe, which quantifies the conservativeness of $\bar{c}$. If this proportion lies within the hyperparameter-controlled range $[p_{\min}, p_{\max}]$, we deem the conservativeness acceptable and adopt $\bar{c}$ as the final cost function. Otherwise, we construct a feedback description L_{feed} from the evaluation outcomes and feed it back into the LLM to guide the regeneration of $\bar{c}$:

$$\bar{c} = \text{LLM}(L_{\text{task}}, L_{\text{cost}}, L_{\text{inst}}, L_{\text{feed}}). \tag{11}$$

If no satisfactory $\bar{c}$ is obtained after exhausting the maximum number of LLM queries, we adopt as the final cost function the candidate that attains the highest accuracy on $\mathcal{D}_{\text{unsafe}}$ and exhibits a conservativeness level on $\mathcal{D}$ closest to the interval $[p_{\min}, p_{\max}]$.

3.3 Overall Algorithm

With the learned $\hat{T}$ and the obtained $\bar{c}$, we can now learn a feasible policy based on the safe-only dataset $\mathcal{D}$. First, we initialize a model rollout dataset $\mathcal{D}_{\hat{T}}$, and use the learning policy π to perform branched rollouts of length H within $\hat{T}$, obtaining a set of branch trajectories $\{\tau_i\}_{i=1}^N$. Each trajectory is of the form $\tau_i = (s_0^i, a_0^i, \bar{c}(s_0^i), \ldots, s_{H-1}^i, a_{H-1}^i, \bar{c}(s_{H-1}^i))$, where $s_0^i \in \mathcal{D}$, $s_t^i \sim \hat{T}(\cdot|s_{t-1}^i, a_{t-1}^i)$ for $t > 0$. Next, for any trajectory τ_i, if it contains a safety violation, i.e., $\sum_{t=0}^{H-1} \bar{c}(s_t^i) > 0$, we add it to $\mathcal{D}_{\hat{T}}$. Otherwise, trajectories without safety violations are discarded. To enhance data diversity during rollouts and thereby improve the efficiency of feasibility identification, we inject Gaussian noise with standard deviation σ_{exp} (a hyperparameter) to the policy actions during rollout. Meanwhile, the existing samples in $\mathcal{D}$ are further re-labeled with costs using $\bar{c}$. Finally, we derive the constraint violation labels according to $h(s) = h_{\min} \leq 0$ if $\bar{c}(s) = 0$, and $h(s) = h_{\max} > 0$ otherwise.

With the constraint violation labels in place, we approximate the minimization operator in the (conservative) feasible Bellman update, i.e., $V_h^*(s') = \min_{a'} Q_h(s', a')$, by employing reverse expectile regression to update the constraint violation value network:

$$\begin{aligned}
\mathcal{L}_{V_h} &= \mathbb{E}_{(s,a)\sim\mathcal{D}\cup\mathcal{D}_{\hat{T}}}[\mathcal{L}_{\text{rev}}^{\tau}(Q_h(s,a) - V_h(s))], \\
\mathcal{L}_{Q_h} &= \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[((1-\gamma)h(s) + \gamma \max(h(s), V_h(s')) - Q_h(s,a))^2] \\
&\quad + \mathbb{E}_{(s,a)\sim\mathcal{D}_{\hat{T}}}[((1-\gamma)h(s) + \gamma \max(h(s), \max_{s'\in\hat{T}(s,a)} V_h(s')) - Q_h(s,a))^2],
\end{aligned} \tag{12}$$

where $\mathcal{L}_{\text{rev}}^{\tau}(u) = |\tau - \mathbb{I}(u > 0)|u^2$, τ is a hyperparameter. Finally, suppose that for a given offline safe RL algorithm, the reward critic learning loss is denoted by $\mathcal{L}_{\text{reward}}$, and the policy learning loss by $\mathcal{L}_{\text{policy}}$. We then directly perform learning using the original offline dataset $\mathcal{D}$

$$\mathcal{L}_{\text{PROCO reward}} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[\mathcal{L}_{\text{reward}}],\ \mathcal{L}_{\text{PROCO policy}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\mathcal{L}_{\text{policy}}(Q_h, V_h)], \tag{13}$$

where $\mathcal{L}_{\text{policy}}(Q_h, V_h)$ denotes the policy learning loss that uses the Q_h and V_h obtained in Equation (12) as the cost critics, and performs policy optimization based on $\mathcal{L}_{\text{policy}}$. We derive the following theorem to show that, even without making any assumptions about model error, integrating PROCO with an arbitrary algorithm does not negatively affect the safety of the resulting policy

Theorem 3.8. *Assume exact lower-quantile aggregation for the feasibility value update, and γ is large enough. For any $\mathcal{D}$ and $\hat{T}$, let $V_{h,origin}$ denote the optimal feasible value function learned by a baseline, and let $V_{h,PROCO}$ denote the optimal feasible value function learned by combining the same baseline with PROCO. Then, for any $s_t \in \mathcal{D}$, we have:*

$$V_{h,PROCO}(s_t) \geq V_{h,origin}(s_t) \text{ or } V_{h,PROCO}(s_t) > 0. \tag{14}$$

Thus, incorporating PROCO into safe policy learning will not degrade safety performance due to underestimation of the safety value, compared to learning without PROCO.

4 Experiments

We evaluate PROCO on 17 Safety-Gymnasium (Ji et al., 2023) tasks from the modified OSRL (Liu et al., 2024) dataset, focusing on four questions: (1) Does PROCO outperform baselines across tasks (Section 4.2)? (2) Do infeasible states in safe-only datasets undermine policy safety, and can PROCO address this (Section 4.3)? (3) How do different components affect performance (Section 4.4)? (4) Can PROCO generalize to datasets with more unsafe samples (Section 4.5)?

4.1 Baselines and Tasks

To evaluate PROCO and its compatibility with different algorithms, we consider three offline safe RL methods: hard-constraint methods **FISOR** (Zheng et al., 2024) and **LSPC** (Koirala et al., 2025), and **CAPS**(IQL) (Chemingui et al., 2025), a soft-constraint method with a similar policy training framework that can be extended to hard constraints. Integrating PROCO yields **FISOR PROCO**, **LSPC PROCO**, and **CAPS PROCO**. We also compare with **BC** and **CDT** (Liu et al., 2023) on safe-only datasets, and **CCAC** (Guo et al., 2025), a soft-constraint method using conservative value estimation. Soft-constraint baselines use a cost limit of 0. Rewards are min-max normalized by the offline dataset, and costs are scaled by 10. Results with normalized cost ≤ 1 are considered safe, with higher reward preferred among safe methods.

We use 17 Safety-Gymnasium tasks for evaluation: 12 navigation tasks covering Point and Car robots across Button, Goal, and Push scenarios (two tasks each), and 5 velocity tasks with Ant, HalfCheetah, Hopper, Swimmer, and Walker2d. We use the most challenging OSRL dataset, whose trajectories exhibit varying constraint violations, and construct safe-only datasets by removing unsafe data while preserving diverse feasibility levels. For each task, 100 unsafe samples are retained to validate LLM-generated cost functions. See Appendix D for details.

4.2 Competitive Results

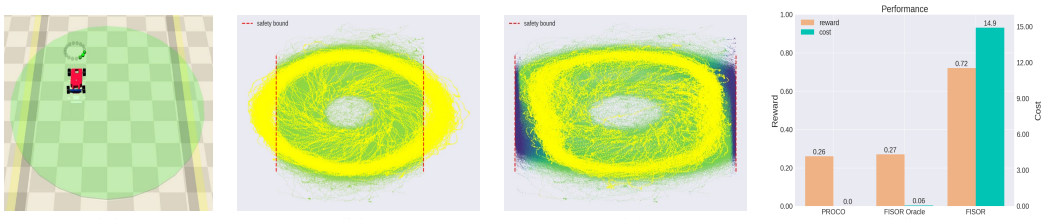
Table 1 summarizes the performance of all methods across Safety-Gymnasium tasks. CCAC shows the weakest safety, indicating that conservative TD learning alone cannot prevent distribution shift and severe violations. BC-based methods are safer by restricting policies to safe-only data, but cannot reliably distinguish feasible from infeasible states. Similarly, SOTA offline safe RL methods-LSPC, CAPS, and FISOR-perform well with abundant unsafe data but incur substantial violations on safe-only datasets as they primarily optimize rewards. In contrast, PROCO identifies feasibility through model-based rollouts and conservative cost functions, improving safety by over 400% across all three variants and by more than 1000% at best. Due to its superior performance, we adopt FISOR PROCO as the default variant, referred to simply as PROCO in subsequent experiments.

4.3 Case Study

Here, we aim to verify whether infeasible states in a safe-only dataset can adversely affect safe policy learning. To this end, we conduct a simple visualization experiment on the Ant Circle task. As illustrated in Figure 2(a), the Circle task requires the agent to move along the circumference of a circle as closely as possible, while crossing the left or right boundaries is considered unsafe.

Table 1: Overall normalized rewards and costs. Each value is averaged over 20 episodes and 3 random seeds. **Bold**: Safe agents. Gray: Unsafe agents. **Blue**: Safe agent with the highest reward.

Task	BC		CDT		CCAC		LSPC		CAPS		FISOR		LSPC PROCO		CAPS PROCO		FISOR PROCO	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.13	5.20	0.47	13.27	0.00	6.54	0.16	4.37	0.13	3.37	0.49	11.50	0.09	1.28	-0.03	1.13	0.01	1.15
PointButton2	0.28	6.68	0.59	15.27	0.61	17.87	0.32	10.04	0.18	4.90	0.46	14.69	0.14	3.02	**0.02**	**0.63**	0.08	2.76
PointGoal1	0.58	3.30	0.75	5.01	0.00	5.29	0.71	3.10	0.61	3.49	0.73	5.70	**0.21**	**0.23**	**0.02**	**0.00**	**0.35**	**0.96**
PointGoal2	0.53	8.07	0.78	13.90	0.87	14.15	0.59	8.55	0.45	5.81	0.59	17.10	**0.11**	**0.45**	**0.05**	**0.66**	**0.08**	**0.38**
PointPush1	0.24	4.65	0.30	4.65	0.02	1.18	0.17	2.21	0.19	3.96	0.34	3.70	0.16	2.09	**0.11**	**0.26**	**0.17**	**0.86**
PointPush2	0.22	3.70	0.28	4.51	-0.08	5.95	0.13	5.76	0.18	3.58	0.27	6.61	0.15	1.79	0.09	1.61	**0.12**	**0.30**
CarButton1	-0.09	3.32	0.25	15.53	0.03	17.26	-0.05	6.51	0.06	4.51	0.47	28.01	-0.03	1.92	-0.02	2.02	**-0.03**	**0.60**
CarButton2	-0.09	4.64	0.35	19.98	0.01	11.94	-0.17	2.49	-0.12	4.02	0.57	25.96	-0.11	1.88	**-0.09**	**0.34**	**0.00**	**0.99**
CarGoal1	0.36	2.15	0.71	5.04	**0.00**	**0.00**	0.47	1.58	0.43	2.15	0.75	4.10	**0.20**	**0.68**	**0.14**	**0.14**	**0.13**	**0.02**
CarGoal2	0.22	2.96	0.67	11.43	0.48	14.73	0.25	3.85	0.18	3.41	0.80	14.56	**0.18**	**0.52**	**0.07**	**0.30**	**0.03**	**0.52**
CarPush1	0.19	1.10	0.32	2.86	-1.51	0.64	**0.19**	**0.91**	0.18	2.39	0.42	1.91	**0.10**	**0.55**	**0.16**	**0.20**	**0.16**	**0.93**
CarPush2	0.08	4.14	0.23	9.88	0.06	3.02	0.09	3.63	0.11	3.67	0.39	8.04	0.12	1.30	0.03	1.50	**0.01**	**0.04**
SwimmerVelocityV1	0.46	5.70	0.70	4.59	0.22	9.40	0.62	6.75	0.51	7.98	-0.05	4.51	**0.05**	**0.58**	**0.03**	**0.08**	**0.02**	**0.12**
HopperVelocityV1	0.35	1.33	0.63	3.44	0.08	7.00	0.08	2.19	0.66	8.22	0.07	3.23	0.17	5.86	**0.26**	**0.06**	**0.11**	**0.07**
HalfCheetahVelocityV1	0.95	4.27	0.95	1.42	1.51	97.03	1.03	45.52	0.98	9.17	1.03	16.13	**0.41**	**0.00**	**0.00**	**0.00**	**0.48**	**0.00**
Walker2dVelocityV1	0.69	2.67	**0.79**	**0.55**	0.05	1.42	**0.80**	**0.00**	**0.78**	**0.39**	0.15	5.71	0.50	2.08	0.53	2.83	**0.09**	**0.58**
AntVelocityV1	0.98	11.54	0.99	1.85	**-1.01**	**0.00**	0.99	11.44	0.98	13.25	1.01	10.47	**0.81**	**0.16**	**0.50**	**0.00**	**0.52**	**0.00**
Average	0.36	4.44	0.57	7.84	0.08	12.55	0.38	6.99	0.38	4.96	0.50	10.70	0.19	1.44	**0.11**	**0.69**	**0.14**	**0.60**
Safe Percent	0/17		1/17		3/17		2/17		1/17		0/17		8/17		12/17		15/17	
Safety Improvement	/		/		/		/		/		/		555%		427%		1011%	



(a) (b) (c) (d)

Figure 2: (a) Visualization of the Circle task. (b) and (c) Visualization of FISOR and PROCO performance, where the red dashed line denotes the safety bound, while the yellow dots indicate the state visitation distribution of the policy. The remaining points represent the value estimations of the constraint violation value function for samples in $\mathcal{D}$, where darker colors correspond to higher estimated violation values. (d) Final performance comparison of FISOR and PROCO.

We compare FISOR with PROCO in Figure 2. Without unsafe samples, FISOR cannot distinguish feasible from infeasible offline samples through its constraint violation value function (Figure 2(b)). The resulting reward-driven policy deviates from the offline distribution even with BC-based extraction, producing many OOD unsafe samples. In contrast, PROCO combines conservative cost estimation with model-based rollouts to identify near-boundary samples as infeasible (Figure 2(c)), allowing timely policy adjustment before leaving the safe region. Consequently, PROCO achieves zero safety violations and performance comparable to **FISOR Oracle**, which uses abundant additional unsafe samples, while FISOR exhibits substantial unsafe behavior (Figure 2(d)).

4.4 Ablation and Analysis Studies

In this section, we first conduct ablation studies to evaluate the contributions of key components in PROCO. The following variants are considered: (1) **W/o Model** omits the use of a dynamics model for generating unsafe samples, instead solely applying the conservative cost function to relabel the offline dataset. (2) **Full Model** utilizes all model rollout data to update all the value functions and the policy. (3) **W/o Relabel** applies the conservative cost function only to label the model rollout data and does not relabel the offline dataset. (4) **Det. Rollout** executes model rollouts deterministically with the policy, omitting the injection of extra noise. (5) **W/o Consv.** does not require the LLM to generate a cost function that is more conservative than the given safety constraint description. (6) **W/o Refl.** does not use the additional unsafe samples $\mathcal{D}_{\text{unsafe}}$ or the safe offline dataset to validate and provide feedback on the correctness and conservativeness of the LLM-generated cost function. As shown in Figure 4(a), when the dynamics model is not used, safety performance drops significantly, confirming the effectiveness of model-based feasibility identification. On the other hand, employing all model rollout data for updates to all value functions and the policy preserves high safety performance but significantly reduces reward performance, suggesting that errors accumulated during model rollouts can destabilize value learning and negatively impact reward estimation. Meanwhile,

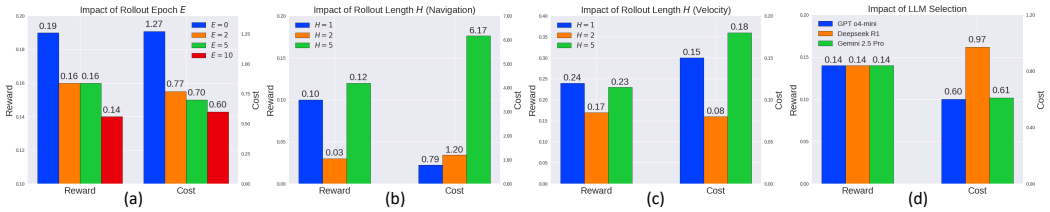


Figure 3: (a) Analysis on rollout epoch E. (b), (c) Analysis on rollout length H. (d) Analysis on LLM selection.

the decline in safety performance for W/o Relabel and Det. Rollout confirms the effectiveness of relabeling offline data with the conservative cost function and adding extra noise during model rollouts to enhance data diversity. Finally, the poor safety performance of W/o Consv. and W/o Refl. highlights the importance of generating a more conservative cost function for effective feasibility identification, as well as the necessity of validating and refining the LLM-generated cost function using existing data. More detailed results are provided in Appendix E.

Next, we evaluate the accuracy of the feasibility identification learned by PROCO. Specifically, we compare the feasible values predicted by PROCO with those produced by the FISOR Oracle, and categorize the results into three types: **Type 1**, where both methods yield the same sign; **Type 2**, where PROCO classifies a state as safe while the FISOR Oracle classifies it as unsafe; and **Type 3**, where PROCO classifies a state as unsafe while the FISOR Oracle classifies it as safe. The results are shown in Figure 4(b). We observe that PROCO achieves a high level of accuracy in feasibility identification, producing predictions that are comparable to those of the FISOR Oracle, even in the absence of unsafe samples. Moreover, Type 2 cases are almost nonexistent, indicating that PROCO exhibits extremely high precision in identifying infeasible samples, further validating the effectiveness of the proposed model-based framework.

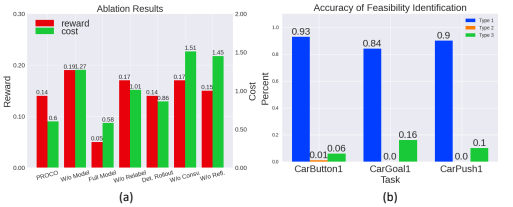


Figure 4: (a) Ablation study results. (b) Feasibility identification accuracy analysis.

Then, we investigate the impact of different hyperparameters on PROCO's performance. During rollouts, three hyperparameters play a crucial role: rollout batch size b, the number of rollout epochs E, and rollout length H. Specifically, b denotes the number of samples drawn, H is the number of steps rolled out per sample, and E indicates how many times the previous process is repeated per rollout. Since both b and E determine the total amount of rollout data, we focus our analysis on E and H. First, as shown in Figure 3(a), increasing E leads to a noticeable decline in reward and cost. This indicates that a larger amount of rollout data enables more accurate cost value learning, further confirming the effectiveness of model-based feasibility identification. Second, Figure 3(b) and (c) show that in Velocity tasks, where the dynamics model is relatively accurate, increasing H does not noticeably affect policy performance. In contrast, in Navigation tasks, where model learning is more challenging, larger values of H lead to a certain degradation in safety performance. These results confirm the necessity of reducing H while employing a conservative cost function when an accurate dynamics model is difficult to learn. In the end, Figure 3(d) presents the effect of different LLMs on performance. GPT o4-mini and Gemini 2.5 Pro demonstrate comparable overall results, whereas Deepseek R1 shows lower safety performance. This suggests that the choice of LLM can impact PROCO's effectiveness, with more capable LLMs generally yielding better outcomes.

4.5 Performance with Different Unsafe Data Number

Finally, we evaluate PROCO and FISOR under varying amounts of unsafe data (10%, 20%, 50%, 100% of the OSRL unsafe dataset) across 17 tasks. The results are shown in Figure 5. We observe that as the amount of unsafe data increases, the reward performance of PROCO remains largely stable, while the cost exhibits a moderate increase. This is likely because incorporating more unsafe

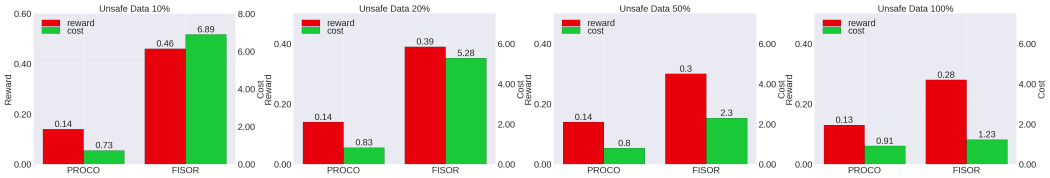


Figure 5: Performance analysis under different unsafe data percent.

samples into the BC term introduces a stronger bias toward unsafe actions. Meanwhile, these results suggest that model errors have a limited impact when the rollout length is small. Importantly, when unsafe data are scarce, PROCO consistently achieves substantially better safety performance than FISOR. Only when the unsafe data reach 100% does the use of PROCO become a trade-off: incorporating PROCO sacrifices some reward performance in exchange for improved safety, whereas omitting it prioritizes reward optimization.

5 Related Work

Safe RL. Safe RL seeks to maximize reward while ensuring safety, commonly modeled as a Constrained Markov Decision Process (CMDP) (Altman, 2021) and solved via constrained optimization (Garcıa & Fernández, 2015; Gu et al., 2024). A widely adopted approach is Lagrangian-based methods, which learn a cost value function with an adaptive multiplier to enforce safety (Stooke et al., 2020). For hard, state-wise constraints, Hamilton-Jacobi (HJ) reachability analysis (Bansal et al., 2017) has been applied to learn cost functions and enforce stricter safety (Yu et al., 2022; Ganai et al., 2023). However, both approaches require unsafe real-world interactions, which limit their practicality. To overcome this limitation, recent research has shifted toward offline safe RL, learning safe policies from pre-collected data to avoid unsafe exploration. Certain approaches integrate conservative value estimation into cost value function learning to counteract cost value underestimation (Xu et al., 2022; Guo et al., 2025), whereas others employ BC-based policy learning, including Decision Transformer (Liu et al., 2023; Xue et al., 2025), DICE (Lee et al., 2022), and IQL (Zheng et al., 2024; Koirala et al., 2025; Zifan et al., 2025), to more effectively address extrapolation errors (Fujimoto et al., 2019) in safe RL.

LLM-assisted decision-making. Leveraging the powerful information processing and reasoning capabilities of LLMs, they have recently been widely adopted in RL (Cao et al., 2024). One prominent line of work explores the use of LLMs for direct decision-making by generating actions or high-level plans conditioned on observations (Yao et al., 2023; Shinn et al., 2023; Prasad et al., 2023). Yet, these methods are generally applicable only to high-level, highly abstract decision-making tasks. A more broadly adopted line of work in RL involves reward function generation, in which LLMs are employed to directly construct reward functions that support skill discovery (Yu et al., 2023), policy learning (Song et al., 2023; Ma et al., 2024; Xie et al., 2024), exploration (Triantafyllidis et al., 2024), or teammate generation (Li et al., 2025a). However, the use of LLMs for cost generalization in safe RL remains largely underexplored. More related work can be seen in Appendix B.

6 Final Remarks

In this work, we propose PROCO, a novel algorithm for learning safe policies offline when unsafe samples are scarce or entirely absent. PROCO first leverages the offline dataset to learn a dynamics model, while simultaneously employing an LLM to generate a conservative cost function tailored to the task. Based on the learned dynamics model and cost function, it then performs branched rollouts from the offline data samples to simulate their potential future evolutions. The generated rollout data are further incorporated into HJ reachability analysis for feasibility identification, which in turn guides policy extraction to ensure both safety and effectiveness. Extensive experiments across diverse tasks demonstrate the superior safety performance of PROCO. In the future, leveraging more powerful Vision-Language Models to extend PROCO to visual tasks and embodied intelligence in a more cost-efficient manner represents a highly promising research direction.

AI USE STATEMENT

In this work, we used generative AI tools as an integral part of our proposed method. During manuscript preparation, we also used generative AI tools to assist with language refinement and proofreading. We did not use generative AI tools to directly generate the manuscript content or references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide complete and executable code in the supplementary materials. The required OSRL datasets are fully open-source, and detailed implementation settings, including hyperparameters, are provided in Appendix C. Therefore, the reported results can be readily reproduced.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

Ahmad Alismail and Carsten Lanquillon. A survey of llm-based methods for synthetic data generation and the rise of agentic workflows. In *International Conference on Human-Computer Interaction*, pp. 119–135. Springer, 2025.

Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253, 2017.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Yassine Chemingui, Aryan Deshwal, Honghao Wei, Alan Fern, and Jana Doppa. Constraint-adaptive policy switching for offline safe reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15722–15730, 2025.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: reinforcement learning via sequence modeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pp. 15084–15097, 2021.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 53945–53968, 2024.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kiante Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. *arXiv preprint arXiv:2505.22866*, 2025.

Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation*, pp. 8550–8556, 2019.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062, 2019.

Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 69764–69797, 2023.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Zhiqi Ge, Hongzhe Huang, Mingze Zhou, Juncheng Li, Guoming Wang, Siliang Tang, and Yueting Zhuang. Worldgpt: Empowering llm as multimodal world model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 7346–7355, 2024.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Zijian Guo, Weichao Zhou, Shengao Wang, and Wenchao Li. Constraint-conditioned actor-critic for offline safe reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: model-based policy optimization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 12519–12530, 2019.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915, 2022.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Juntao Dai, and Yaodong Yang. Safety-gymnasium: a unified safe reinforcemei learning benchmark. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 18964–18993, 2023.

Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 12140–12147. IEEE, 2024.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: model-based offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 21810–21823, 2020.

Prajwal Koirala, Zhanhong Jiang, Soumik Sarkar, and Cody Fleming. Latent safety-constrained policy approach for safe offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 1179–1191, 2020.

Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. In *International Conference on Learning Representations*, 2022.

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

Lihe Li, Lei Yuan, Pengsen Liu, Tao Jiang, and Yang Yu. Llm-assisted semantically diverse teammate generation for efficient multi-agent coordination. In *Forty-second International Conference on Machine Learning*, 2025a.

Yinchuan Li, Xinyu Shao, Jianping Zhang, Haozhi Wang, Leo Maxime Brunswic, Kaiwen Zhou, Jiqian Dong, Kaiyang Guo, Xiu Li, Zhitang Chen, et al. Generative models in decision making: A survey. *arXiv preprint arXiv:2502.17100*, 2025b.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International conference on machine learning*, pp. 21611–21630, 2023.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, and Ding Zhao. Datasets and benchmarks for offline safe reinforcement learning. *Journal of Data-centric Machine Learning Research*, 2024.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *arXiv preprint arXiv:2506.12811*, 2025.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Jing Nie, Jiachen Jiang, Yang Li, Huting Wang, Sezai Ercisli, and Linze Lv. Data and domain knowledge dual-driven artificial intelligence: Survey, applications, and challenges. *Expert Systems*, 42 (1):e13425, 2025.

Fabian Paischer, Thomas Adler, Vihang Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In *International Conference on Machine Learning*, pp. 17156–17185, 2022.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*, 2023.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 8634–8652, 2023.

Jiayang Song, Zhehua Zhou, Jiawei Liu, Chunrong Fang, Zhan Shu, and Lei Ma. Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics. *arXiv preprint arXiv:2309.06687*, 2023.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143, 2020.

Yihao Sun, Jiaji Zhang, Chengxing Jia, Haoxin Lin, Junyin Ye, and Yang Yu. Model-bellman inconsistency for model-based offline reinforcement learning. In *International Conference on Machine Learning*, pp. 33177–33194, 2023.

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, 2012.

Eleftherios Triantafyllidis, Filippos Christianos, and Zhibin Li. Intrinsic language-guided exploration for complex long-horizon robotic manipulation tasks. In *2024 IEEE International Conference on Robotics and Automation*, pp. 7493–7500, 2024.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Ruiqi Xue, Ziqian Zhang, Lihe Li, Cong Guan, Lei Yuan, and Yang Yu. Adaptable safe policy learning from multi-task data with constraint prioritized decision transformer. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Xue Yan, Yan Song, Xidong Feng, Mengyue Yang, Haifeng Zhang, Haitham Bou Ammar, and Jun Wang. Efficient reinforcement learning with large language model priors. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: implicit constraint approach for offline multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pp. 10299–10312, 2021.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International conference on machine learning*, pp. 25636–25655, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: model-based offline policy optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 14129–14142, 2020.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. In *7th Annual Conference on Robot Learning*, 2023.

Lixian Zhang, Ruixian Zhang, Tong Wu, Rui Weng, Minghao Han, and Ye Zhao. Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles. *IEEE transactions on neural networks and learning systems*, 32(12):5435–5444, 2021.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations*, 2024.

LIU Zifan, Xinran Li, and Jun Zhang. C2iql: Constraint-conditioned implicit q-learning for safe offline reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

A Mathematical Proofs

A.1 Proof of Proposition 3.1

Proof. First, we aim to proof that

$$|\max(a_1, b_1) - \max(a_2, b_2)| \leq \max(|a_1 - a_2|, |b_1 - b_2|). \tag{15}$$

To start, if $a_1 \geq b_1$, then

$$\max(a_1, b_1) - \max(a_2, b_2) = a_1 - \max(a_2, b_2) \leq a_1 - a_2. \tag{16}$$

Otherwise, if $b_1 \geq a_1$, then

$$\max(a_1, b_1) - \max(a_2, b_2) = b_1 - \max(a_2, b_2) \leq b_1 - b_2. \tag{17}$$

Therefore, we have

$$\max(a_1, b_1) - \max(a_2, b_2) \leq \max(a_1 - a_2, b_1 - b_2). \tag{18}$$

Similarly, by switching the subscript 1 and 2, we have

$$\max(a_2, b_2) - \max(a_1, b_1) \leq \max(a_2 - a_1, b_2 - b_1), \tag{19}$$

which is equivalent to

$$-(\max(a_1, b_1) - \max(a_2, b_2)) \leq \max(-(a_1 - a_2), -(b_1 - b_2)). \tag{20}$$

By observing that $\max(x, y) \leq \max(|x|, |y|)$, we have

$$-\max(|a_1 - a_2|, |b_1 - b_2|) \leq \max(a_1, b_1) - \max(a_2, b_2) \leq \max(|a_1 - a_2|, |b_1 - b_2|), \tag{21}$$

which means

$$|\max(a_1, b_1) - \max(a_2, b_2)| \leq \max(|a_1 - a_2|, |b_1 - b_2|). \tag{22}$$

Therefore, we have

$$\begin{aligned}
||\bar{\mathcal{B}}^* Q - \bar{\mathcal{B}}^* Q'||_\infty &= \sup_{s,a} |\bar{\mathcal{B}}^* Q(s,a) - \bar{\mathcal{B}}^* Q'(s,a)| && (23)\\
&= \gamma \sup_{s,a} |\max\{h(s), \max_{s' \in \hat{T}(s,a)} \min_{a'} Q(s', a')\} \\
&\quad - \max\{h(s), \max_{s' \in \hat{T}(s,a)} \min_{a'} Q'(s', a')\}| && (24)\\
&\leq \gamma \sup_{s,a} |\max_{s' \in \hat{T}(s,a)} \min_{a'} Q(s', a') - \max_{s' \in \hat{T}(s,a)} \min_{a'} Q'(s', a')| && (25)\\
&\leq \gamma \sup_{s,a} \max_{s' \in \hat{T}(s,a)} \max_{a'} |Q(s', a') - Q'(s', a')| && (26)\\
&\leq \gamma \sup_{s',a'} |Q(s', a') - Q'(s', a')| && (27)\\
&= \gamma ||Q - Q'||_\infty, && (28)
\end{aligned}$$

where inequality 25 owns to inequality 22. Thus, $\bar{\mathcal{B}}^*$ is a γ contraction mapping in the ∞-norm. Then, for any Q, Q', if $Q \geq Q'$ pointwise, then

$$\begin{aligned}\bar{\mathcal{B}}^* Q &= (1-\gamma)h(s) + \gamma \max\{h(s), \max_{s' \in \hat{T}(s,a)} \min_{a'} Q(s', a')\} \\ &\geq (1-\gamma)h(s) + \gamma \max\{h(s), \min_{a'} Q(s', a')\}, s' \sim T(s,a) \\ &\geq (1-\gamma)h(s) + \gamma \max\{h(s), \min_{a'} Q'(s', a')\}, s' \sim T(s,a) \\ &= \mathcal{B}^* Q'.\end{aligned} \tag{29}$$

Now, let Q^0 be any initial Q function, and define $\bar{Q}^k = (\bar{\mathcal{B}}^*)^k Q_0, Q^k = (\mathcal{B}^*)^k Q_0$. Then, we have $\bar{Q}^k \geq Q^k, \forall k \in \mathbb{N}$. Therefore, taking the limits $\bar{Q}^*_h = \lim_{k\to\infty} \bar{Q}^k$ and $Q^*_{h,T} = \lim_{k\to\infty} Q^k$, we obtain $\bar{Q}^*_h(s,a) \geq Q^*_{h,T}(s,a)$ pointwise. □

A.2 Proof of Theorem 3.4

Lemma A.1. *Suppose that Assumption 3.2 holds, then $\forall s \in \bar{S}^*_f = S - S^*_f$, it holds that $\forall a, Q^*_h(s,a) \geq h_{min} + \gamma^{H^*}(h_{max} - h_{min})$, where Q^*_h is the learned action-value function by the feasible Bellman operator, and h_{min} is the minimum value of $h(s)$, and h_{max} is the value of $h(s)$ when it is unsafe, that is whenever $h(s) > 0$, $h(s) = h_{max}$.*

Proof. For any trajectory τ starting from the infeasible state s_0, suppose it first violates at horizon t, that is for $(s_0, a_0, s_1, a_1, \ldots, s_t)$, $h(s_t) = h_{\max}$. Therefore, we have

$$\begin{aligned}Q^*_h(s_t, a_t) &= (1-\gamma)h(s_t) + \gamma \max\{h(s_t), \min_{a'} Q^*_h(s_{t+1}, a_{t+1})\} \\ &\geq (1-\gamma)h_{\max} + \gamma h_{\max} = h_{\max}.\end{aligned} \tag{30}$$

Similarly, by recursion we have

$$\begin{aligned}Q^*_h(s_{t-k}, a_{t-k}) &\geq (1-\gamma)h_{\min}(1 + \gamma + \cdots + \gamma^{k-1}) + \gamma^k h_{\max} \\ &= h_{\min} + \gamma^k(h_{\max} - h_{\min}).\end{aligned} \tag{31}$$

For that $\gamma < 1$ and $h_{\max} - h_{\min} > 0$, and the arbitrariness of trajectory τ, we get

$$\forall a_0, Q^*_h(s_0, a_0) \geq h_{\min} + \gamma^t(h_{\max} - h_{\min}) \geq h_{\min} + \gamma^{H^*}(h_{\max} - h_{\min}). \tag{32}$$

□

Based on Lemma A.1, we can now proceed to the proof of Theorem 3.4.

Proof. For any infeasible state s_0, suppose $|\hat{T}| = N$, that is the ensemble dynamics model set has N dynamics models, using the following rollout method

- Using each dynamics model in $\hat{T}$ to obtain the fist next state set $\{s\}_1$ with N states.
- For each state in $\{s\}_1$, using each dynamics model to generate next rollout state set.
- Repeat the above two step, untill reaching horizon H^*.

This will generate N^{H^*} rollout branches, and due to Assumption 3.2 and Assumption 3.3, there will be at least one branch that is unsafe. Therefore, if there is only one unsafe branch, then this will be the ground truth branch, and according to Lemma A.1, it can be ensured that $\forall a, Q^*_h(s_0, a) \geq h_{\min} + \gamma^{H^*}(h_{\max} - h_{\min})$. Otherwise, if there are more than one unsafe branch, then we don't need to figure out the ground truth branch, choosing any one of them to update the action value function can also lead to $\forall a, Q^*_h(s_0, a) \geq h_{\min} + \gamma^{H^*}(h_{\max} - h_{\min})$. Then, according to Proposition 3.1 and selecting $\gamma \in (\sqrt[H^*]{\frac{h_{\min}}{h_{\min} - h_{\max}}}, 1)$, we have

$$\forall a, \bar{Q}^*_h(s_0, a) \geq Q^*_h(s_0, a) \geq h_{\min} + \gamma^{H^*}(h_{\max} - h_{\min}) > 0. \tag{33}$$

□

A.3 Proof of Theorem 3.7

Lemma A.2. *(Janner et al., 2019) Suppose the expected KL-divergence between two transition distributions is bounded as* $\max_t \mathbb{E}_{s\sim p_1^t(s)} D_{KL}(p_1(s'|s)||p_2(s'|s)) \leq \delta$*, and the initial stat distributions are the same* $p_1^0(s) = p_2^0(s)$*. Then the distance in the state marginal is bounded as*

$$D_{TV}(p_1^t(s)||p_2^t(s)) \leq t\delta. \tag{34}$$

Proof. Please refer to Janner et al. (2019). □

Based on Lemma A.2, we can now proceed to the proof of Theorem 3.7.

Proof. First, according to Assumption 3.5, Assumption 3.6 and Lemma A.2, we have $|\mathbb{E}_{p_1^t}[h(s)] - \mathbb{E}_{p_2^t}[h(s)]| \leq tK\delta$, which means $\mathbb{E}_{p_2^t}[h(s)] \geq \mathbb{E}_{p_1^t}[h(s)] - tK\delta$ holds for any t. Meanwhile, similar to the proof of Lemma 1, it is clear that $Q_h^{p_2}(s,a) \geq h_{\min} + \gamma^t(\mathbb{E}_{p_2^t}[h(s)] - h_{\min})$, where $Q_h^{p_2}$ refers to the learned value function with the feasible Bellman operator and state transition distribution p_2. (A critical observation is $\forall a, \mathbb{E}_{s\sim p_2^t}[Q(s,a)] \geq \mathbb{E}_{s\sim p_2^t}[h(s)]$.)

Therefore, combining the results before and we can get $Q_h^{p_2}(s,a) \geq (1-\gamma^t)h_{\min} + \gamma^t\mathbb{E}_{p_1^t}[h(s)] - \gamma^t tK\delta$ holds for any t, which means

$$Q_h^{p_2}(s,a) \geq \max_{t\in\{1,\dots,H^*\}} [(1-\gamma^t)h_{\min} + \gamma^t\mathbb{E}_{p_1^t}[h(s)] - \gamma^t tK\delta]. \tag{35}$$

Finally, apply Proposition 3.1, we have

$$\begin{aligned} \bar{Q}_h^*(s,a) \geq Q_h^{p_2}(s,a) &\geq \max_{t\in\{1,\dots,H^*\}} [(1-\gamma^t)h_{\min} + \gamma^t\mathbb{E}_{p_1^t}[h(s)] - \gamma^t tK\delta] \\ &\geq (1-\gamma^{t^*})h_{\min} + \gamma^{t^*}\mathbb{E}_{p_1^{t^*}}[h(s)] - \gamma^{t^*}t^*K\delta, \end{aligned} \tag{36}$$

where $t^* = \arg\max_{t\in\{1,\dots,H^*\}}[\mathbb{E}_{p_1^t}[h(s)]]$. □

A.4 Proof of Theorem 3.8

Lemma A.3. *Let* $x = \{x_1, x_2, \dots, x_n\}$*, and let* y_{old} *denote the* τ*-quantile of* x*. Suppose we augment* x *by adding an arbitrary value* $\alpha > 0$*, and let the resulting* τ*-quantile be denoted by* y_{new}*. Then, if* $y_{old} > 0$*, it necessarily follows that* $y_{new} > 0$*.*

Proof. Let us define the empirical cumulative distribution function (ECDF). For a given sample set $x = \{x_1, x_2, \dots, x_n\}$, the empirical distribution function F_n is defined as

$$F_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{x_i \leq t\}}, \tag{37}$$

where $\mathbf{1}_{\{x_i\leq t\}}$ is the indicator function that equals 1 if $x_i \leq t$ and 0 otherwise.

Since $y_{\text{old}} > 0$, then we have

$$F_n(0) < \tau. \tag{38}$$

After adding a positive value $\alpha > 0$ to x, the number of sample points in $(-\infty, 0]$ remains unchanged. Let m denote the original number of samples satisfying $x_i \leq 0$, then

$$F_n(0) = \frac{m}{n} < \tau. \tag{39}$$

In the new sample set, we have

$$F_{n+1}(0) = \frac{m}{n+1} \leq \frac{m}{n} < \tau. \tag{40}$$

Therefore, we have

$$y_{\text{new}} = \inf\{t : F_{n+1}(t) \geq \tau\} > 0. \tag{41}$$

□

Lemma A.4. *Let $x = \{x_1, x_2, \ldots, x_n\}$, and let y_{old} denote the τ-quantile of x. Suppose we change any $x_t > 0$ to another positive value $x'_t > 0$, and let the resulting τ-quantile be denoted by y_{new}. Then, if $y_{old} > 0$, it necessarily follows that $y_{new} > 0$.*

Proof. Similarly like above, since $y_{\text{old}} > 0$, then we have

$$F_n(0) < \tau. \tag{42}$$

After changing a positive value $x_t > 0$ to $x'_t > 0$, the number of sample points in $(-\infty, 0]$ remains unchanged. Let m denote the original number of samples satisfying $x_i \leq 0$, then

$$F_n(0) = \frac{m}{n} < \tau. \tag{43}$$

In the new sample set, we have

$$F'_n(0) = \frac{m}{n} = F_n(0) < \tau. \tag{44}$$

Therefore, we have

$$y_{\text{new}} = \inf\{t : F'_n(t) \geq \tau\} > 0. \tag{45}$$

□

Proof. Let's begin the proof of Theorem 3.8. Without loss of generality, let $V_{r,\text{orgin}}, V_{h,\text{orgin}}$ denote the reward and feasible value functions obtained without PROCO, respectively, and let $V_{r,\text{PROCO}}, V_{h,\text{PROCO}}$ denote the corresponding value functions obtained with PROCO. First, the reward value update rule remains unchanged after incorporating PROCO. Therefore, $\forall s_t$, we have

$$V_{r,\text{PROCO}}(s_t) = V_{r,\text{orgin}}(s_t). \tag{46}$$

We analyze the idealized quantile update, with reverse expectile regression in Eq. (12) serving as its practical approximation as used in (Zheng et al., 2024). Under this assumption, V_h is a lower quantile of the critic values. For any s_t in $\mathcal{D}$, if none of the future states along its trajectory produces an unsafe branched rollout, then $V_h(s_t)$ is not affected by PROCO, $V_{h,\text{PROCO}}(s_t) = V_{h,\text{orgin}}(s_t)$. Otherwise, suppose that at some future state s_{t+k}, an unsafe branched rollout occurs, whose first action is denoted by a'. Analogous to the derivation in Lemma A.1, as long as the discount factor γ is sufficiently large, we can ensure that $Q_h(s_{t+k}, a') > 0$. In this case, the quantile regression for $V_h(s_{t+k})$ can be viewed as augmenting its fitting dataset with an additional value greater than zero. We now consider two possible cases

- When $V_{h,\text{orgin}}(s_{t+k}) <= 0$, the inclusion of $Q_h(s_{t+k}, a')$ clearly increases the fitted value of $V_h(s_{t+k})$. By propagating this effect backward from time step $t + k$, we obtain

$$V_{h,\text{PROCO}}(s_t) \geq V_{h,\text{orgin}}(s_t). \tag{47}$$

- When $V_{h,\text{orgin}}(s_{t+k}) > 0$, we first apply Lemma A.3, which guarantees that incorporating $Q_h(s_{t+k}, a')$ still preserves $V_h(s_{t+k}) > 0$. Next, we propagate this effect backward from time step $t + k$. If there exists some $t' \in [t, t + k)$, such that $V_{h,\text{orgin}}(s_{t'}) <= 0$, then the above modification does not affect the safety value in timestep t'. This is because, for a dataset whose fitted quantile is less than or equal to zero, replacing one positive value with another positive value does not change the resulting quantile. Consequently, further backward propagation from t' to t also has no effect, yielding

$$V_{h,\text{PROCO}}(s_t) = V_{h,\text{orgin}}(s_t). \tag{48}$$

  Otherwise, if there exists no $t' \in [t, t + k)$ such that $V_{h,\text{orgin}}(s_{t'}) <= 0$, then during the backward propagation from $t + k$ to t, we can repeatedly apply Lemma A.4, which yields

$$V_{h,\text{PROCO}}(s_t) > 0. \tag{49}$$

Therefore, for all $s_t \in \mathcal{D}$, we have

$$V_{h,\text{PROCO}}(s_t) \geq V_{h,\text{orgin}}(s_t) \text{ or } V_{h,\text{PROCO}}(s_t) > 0. \tag{50}$$

Taken together, and noting that policy updates rely solely on the original offline dataset $\mathcal{D}$, incorporating PROCO into safe policy learning neither affects the reward value estimation of the policy nor degrades safety performance due to underestimation of the safety value, compared to learning without PROCO. □

Remark. Theorem 3.8 indicates that PROCO can be integrated with existing offline safe reinforcement learning algorithms without making any assumptions about the number of unsafe samples in the dataset. Moreover, regardless of the magnitude of the dynamics model error, such integration does not lead to degraded policy safety due to the underestimation of the feasible value function. The accuracy of the dynamics model primarily influences the types of samples that are conservatively evaluated in real time. When the dynamics model is sufficiently accurate, the samples for which PROCO induces more conservative decisions tend to correspond to genuinely infeasible or risky states. In contrast, when the dynamics model is less accurate, the method may behave conservatively on some feasible samples, resulting in overly conservative policies and degraded reward performance, or may fail to apply conservative estimates on infeasible samples, potentially affecting safety performance. Nevertheless, even in cases where safety performance is impacted, the preceding analysis guarantees that the learned feasibility value will not become less conservative than that obtained without PROCO due to safety-value underestimation.

B More Related Work

Offline RL. Offline RL trains policies using pre-collected datasets, avoiding real-world trial and error, which is critical for deploying RL in practical settings. Its primary challenge is addressing the extrapolation error (Prudencio et al., 2023). Methods such as CQL (Kumar et al., 2020) and ICQ (Yang et al., 2021) penalize the value function of unseen actions in order to constrain actions to those observed in the offline dataset, while BCQ (Fujimoto et al., 2019) explicitly restricts candidate actions to remain close to the offline data distribution by leveraging a CVAE. Furthermore, approaches like IQL (Kostrikov et al., 2022) and DT (Chen et al., 2021) adopt BC-based supervised learning to fully confine the policy distribution within the offline dataset distribution, thereby avoiding the effect of extrapolation error. Others, such as MOReL (Kidambi et al., 2020), MOPO (Yu et al., 2020), and MOBILE (Sun et al., 2023), learn the environment models from the offline data and utilize these models with uncertainty estimates to avoid OOD regions with low model accuracy.

Generative model assisted RL. Driven by the powerful capacity of generative models to capture multimodal distributions, as well as their rapid progress in domains like natural language processing, recent years have witnessed a growing interest in applying such models to reinforcement learning (Li et al., 2025b). Apart from the aforementioned applications, LLMs can also serve other roles in RL, such as information representation and processing (Paischer et al., 2022), action space compression (Yan et al., 2025), world modeling (Ge et al., 2024), and multi-agent task allocation (Kannan et al., 2024). In addition to LLMs, other generative models are gaining growing traction in RL. Diffusion models, in particular, have emerged as the most widely applied due to their strong expressive power. They have been utilized for planning (Janner et al., 2022; Ajay et al., 2023), for offline policy learning (Wang et al., 2023; Chi et al., 2023; Ding & Jin, 2024), and more recently for online policy learning (Ding et al., 2024; Celik et al., 2025; Ma et al., 2025). Flow matching, due to its simpler generative process compared to diffusion models, has also recently attracted increasing attention in RL (Park et al., 2025; Espinosa-Dice et al., 2025; Lv et al., 2025).

C Implementation Details

In this section, we will offer the implementation details of PROCO.

C.1 Model Learning

We begin with model learning using the safe-only dataset $\mathcal{D}$. Specifically, each model $T \in \hat{T}$ is formulated as a Gaussian distribution over the next state given the current state and action:

$$T(s'|s,a) = \mathcal{N}(\mu(s,a), \Sigma(s,a)). \tag{51}$$

Then, the ensemble dynamics model is optimized by

$$\max_{\hat{T}} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[\log \hat{T}(s'|s,a)]. \tag{52}$$

In training, seven candidate models T are trained to construct the ensemble dynamics model $\hat{T}$, and the best 5 models are picked based on the validation prediction error on a held-out set that contains

20% samples in $\mathcal{D}$. During model rollout, we randomly pick one dynamics model from the best 5 models to obtain s'. For the cost label, given the cost function $\bar{c}$, to mitigate the impact of model uncertainty and obtain a more conservative estimate, we label a sample as 1 if any s' of the five best models' outputs satisfies $\bar{c}(s') = 1$.

C.2 Conservative Cost Function Generation

The key to LLM-based cost function generation is the formulation of the prompt input. Accordingly, this section details the prompt design employed in our approach.

First, we present the overall instruction used when initially employing the LLM for cost function generation

Instruction

You are an expert python coder. Please help me write some functions to evaluate whether a given observation is safe or not. The function should return 0 if the observation is safe, and 1 otherwise. The function should be called get_cost(observation).
The **cost description** is: {Cost Description}.
The **observation description** is: {Observation Description}.
Now, please give the function get_cost, you should be a little **more conservative**, some situations that do not violate safety constraints but are close to doing so can also be considered unsafe.

Subsequently, we present two examples to demonstrate the Cost Description, Observation Description within the instruction above, and Feedback provided for the generated cost function.

Push2 First, the Cost Description for the Push2 task is composed as follows:

Push2 Cost Description

If you are within 0.3 meters of hazard, it is unsafe. If you contact with a pillar (being very close to the pillar), it is unsafe.

The corresponding Observation Description is as follows:

Push2 Observation Description

The observation contains some agent specific dimensions plus 4 radar vetors, the last 16 dimensions is the push box radar, the 16 dimensions before push box radar is the pillar radar, the 16 dimensions before pillar radar is the hazard radar. Each radar has 16 dimensions, meaning taking the robot's forward direction as 0 degrees, determine whether a detected target is present at every 22.5-degree interval around it. However, there is no information about robot's forward direction in the observation. The max radar detection range is 3 meters. If in a given direction, the detected target is d meters away, and d$\leq$3, then the radar will give the number as (3-d)/3. Meanwhile, since the observation lacks a collision sensor, contact with a target does not result in a radar reading of zero distance. This discrepancy is due to the physical volume of the target, whose radius generally does not exceed 0.3 meters.

After incorporating the above content into the instruction and inputting it into the LLM (GPT o4-mini), it first generated the following cost function:

Push2 Output Function1

```
def get_cost(observation):
    """
    Returns 0 if the observation is safe, 1 if unsafe.
```

```
    Parameters
    ----------
    observation : array-like, length >= 48
        The last 48 entries are three 16-dimensional radars:
          - hazard radar:  obs[-48:-32]
          - pillar radar:  obs[-32:-16]
          - push-box radar: obs[-16:]
        Radar values r correspond to distances d = 3*(1 - r).

    Safety rules (conservative):
      1. Any hazard reading r >= 0.90    unsafe
      2. Any pillar reading r >= 0.85    unsafe

    Returns
    -------
    int
        0 if safe, 1 if unsafe.
    """
    obs = np.asarray(observation)
    if obs.ndim != 1 or obs.size < 48:
        raise ValueError("Observation must be a 1D array with length
        ↪  >= 48.")

    # extract the 16-dim hazard and pillar readings
    hazard_r = obs[-48:-32]
    pillar_r = obs[-32:-16]

    # thresholds
    HAZARD_THRESH = 0.90  # exact 0.3 m
    PILLAR_THRESH = 0.85  # small margin beyond contact

    # check if any reading exceeds threshold
    too_close_to_hazard = np.any(hazard_r >= HAZARD_THRESH)
    contact_with_pillar = np.any(pillar_r >= PILLAR_THRESH)

    return 1 if (too_close_to_hazard or contact_with_pillar) else 0
```

However, upon evaluation, this cost function labeled 2% of the samples in $\mathcal{D}$ as unsafe, which does not meet our conservative requirement of labeling samples within $[p_{\min}, p_{\max}]$ as unsafe, where $p_{\min} = 10\%$ and $p_{\max} = 30\%$ in practice. Therefore, we provided the following feedback:

Push2 Feedback1

For safe samples, this function classifies 2% of them as unsafe. We want to classify 10%-30% safe samples as unsafe. It should be a little more conservative.

Based on the above feedback, the LLM made corresponding modifications to the output function

Push2 Output Function2

```
def get_cost(observation):
    """
    Returns 0 if the observation is safe, 1 if unsafe
    ↪  (conservatively).

    Parameters
    ----------
    observation : array-like, length >= 48
        The last 48 entries are:
          - hazard radar:  obs[-48:-32]
```

```
          - pillar radar:  obs[-32:-16]
          - push-box radar: obs[-16:]

    Conservative thresholds:
      - Hazard: r >= 0.80  d <= 0.60 m (unsafe)
      - Pillar: r >= 0.75  d <= 0.75 m (unsafe)

    Returns
    -------
    int
        0 if safe, 1 if unsafe.
    """
    obs = np.asarray(observation)
    if obs.ndim != 1 or obs.size < 48:
        raise ValueError("Observation must be a 1D array with length
        ↪  >= 48.")

    hazard_r = obs[-48:-32]
    pillar_r = obs[-32:-16]

    HAZARD_THRESH = 0.80   # ~0.60 m
    PILLAR_THRESH = 0.75   # ~0.75 m

    too_close_to_hazard = np.any(hazard_r >= HAZARD_THRESH)
    close_to_pillar = np.any(pillar_r >= PILLAR_THRESH)

    return 1 if (too_close_to_hazard or close_to_pillar) else 0
```

Nevertheless, the revised cost function proved excessively conservative, labeling 40% of safe samples as unsafe. Consequently, we supplied the LLM with additional feedback:

Push2 Feedback2

For safe samples, this function classifies 40% of them as unsafe. We want to classify 10%-30% safe samples as unsafe. It is too conservative.

Finally, the LLM adjusted the cost function once again, yielding:

Push2 Output Function3

```
def get_cost(observation):
    """
    Returns 0 if the observation is safe, 1 if unsafe (moderately
    ↪  conservative).

    Parameters
    ----------
    observation : array-like, length >= 48
        The last 48 entries are:
          - hazard radar:  obs[-48:-32]
          - pillar radar:  obs[-32:-16]
          - push-box radar: obs[-16:]

    Thresholds tuned for 10-30% false positives on safe samples:
      - Hazard: r >= 0.85  d <= 0.45 m (unsafe)
      - Pillar: r >= 0.80  d <= 0.60 m (unsafe)

    Returns
    -------
    int
        0 if safe, 1 if unsafe.
```

```
    """
    obs = np.asarray(observation)
    if obs.ndim != 1 or obs.size < 48:
        raise ValueError("Observation must be a 1D array with length
        ↪  >= 48.")

    hazard_r = obs[-48:-32]
    pillar_r = obs[-32:-16]

    HAZARD_THRESH = 0.85  # ~0.45 m
    PILLAR_THRESH = 0.80  # ~0.60 m

    too_close_to_hazard = np.any(hazard_r >= HAZARD_THRESH)
    close_to_pillar = np.any(pillar_r >= PILLAR_THRESH)

    return 1 if (too_close_to_hazard or close_to_pillar) else 0
```

Walker2dVelocity In MuJoCo Velocity tasks, the agent is deemed unsafe once its instantaneous velocity surpasses a predefined threshold. Nevertheless, the velocity included in the agent's observation reflects only the terminal velocity at the end of a step, which may deviate from the velocity during step execution. Consequently, an LLM-generated cost function may align well with the textual description, or even adopt a more conservative stance, but still fail to achieve perfect classification accuracy on the 100 samples in $\mathcal{D}_{\text{unsafe}}$. This example illustrates such a case. First, the Cost Description for the Walker2dVelocity task is composed as follows:

Walker2dVelocity Cost Description

If the velocity of the robot exeeds 2.3415, it is unsafe. When calculating the velocity, we only consider x-coordinate and y-coordinate (if exists).

The corresponding Observation Description is as follows:

Walker2dVelocity Observation Description

The observation contains 17 dimensions. The first is z-coordinate of the front tip; the second is angle of the torso; the third is angle of the thigh joint; the forth is angle of the leg joint; the fifth is angle of the foot joint; the sixth is angle of the left thigh joint; the seventh is angle of the left leg joint; the eighth is angle of the left foot joint; the nineth is velocity of the x-coordinate of the torso; the tenth is velocity of the z-coordinate (height) of the torso; the eleventh is angular velocity of the angle of the torso; the twelveth is angular velocity of the thigh hinge; the thirteenth is angular velocity of the leg hinge; the forteenth is angular velocity of the foot hinge; the fifteenth is angular velocity of the thigh hinge; the sixteenth is angular velocity of the leg hinge; the seventeenth is angular velocity of the foot hinge.

After incorporating the above content into the instruction and inputting it into the LLM, it first generated the following cost function:

Walker2dVelocity Output Function1

```
def get_cost(observation, max_safe_vel=2.3415, safety_margin=0.05):
    """
    Return 0 if the observation is safe, 1 otherwise.

    We consider only the torso's x-velocity (observation[8]).
    Any |velocity| >= (1 - safety_margin) * max_safe_vel is deemed
    ↪  unsafe.
```

```
    Parameters
    ----------
    observation : sequence of float, length >= 9
        The 17-dimensional observation vector.
    max_safe_vel : float
        The hard safety limit on velocity.
    safety_margin : float
        Fractional margin to be conservative (e.g. 0.05 means 5%
        ↪ below max_safe_vel).

    Returns
    -------
    int
        0 if safe, 1 if unsafe.
    """
    # pick out the x-velocity of the torso
    v_x = observation[8]

    # define the conservative threshold
    threshold = (1.0 - safety_margin) * max_safe_vel

    # unsafe if we're at or above the threshold
    return 1 if abs(v_x) >= threshold else 0
```

However, the accuracy of this function on $\mathcal{D}_{\text{unsafe}}$ is only 98%. Therefore, we provided feedback to the LLM:

Walker2dVelocity Feedback1

For 100 unsafe testing samples, this function achieves 98% accuracy, it should be a little more conservative.

Based on the above feedback, the LLM made corresponding modifications to the output function

Walker2dVelocity Output Function2

```
def get_cost(observation, max_safe_vel=2.3415, safety_margin=0.10):
    """
    Return 0 if the observation is safe, 1 otherwise.

    Unsafe if torso's x-velocity exceeds (1 - safety_margin) *
    ↪ max_safe_vel.
    This version is more conservative, using a 10% margin.

    Parameters
    ----------
    observation : sequence of float, length >= 9
        The 17-dimensional observation vector.
    max_safe_vel : float
        The hard velocity safety threshold.
    safety_margin : float
        A conservative margin to catch near-threshold values.

    Returns
    -------
    int
        0 if safe, 1 if unsafe.
    """
    v_x = observation[8]
    threshold = (1.0 - safety_margin) * max_safe_vel
    return 1 if abs(v_x) >= threshold else 0
```

The accuracy of this function on $\mathcal{D}_{\text{unsafe}}$ remains at only 99%. Therefore, we provided additional feedback

Walker2dVelocity Feedback2

For 100 unsafe testing samples, this function achieves 99% accuracy, it should be a little more conservative.

Finally, the LLM adjusted the cost function once again, yielding:

Walker2dVelocity Output Function3

```python
def get_cost(observation, max_safe_vel=2.3415, safety_margin=0.15):
    """
    Return 0 if the observation is safe, 1 otherwise.

    Unsafe if torso's x-velocity magnitude exceeds
    (1 - safety_margin) * max_safe_vel.

    This default uses a 15% margin (i.e. flags |v_x| >= ~1.9903).

    Parameters
    ----------
    observation : sequence of float, len >= 9
        The 17-dimensional observation vector.
    max_safe_vel : float
        The hard velocity safety limit.
    safety_margin : float
        Fractional buffer under max_safe_vel (default 0.15).

    Returns
    -------
    int
        0 if safe, 1 if unsafe.
    """
    # torso x-velocity is at index 8
    v_x = observation[8]

    # build a conservative threshold
    threshold = (1.0 - safety_margin) * max_safe_vel

    return 1 if abs(v_x) >= threshold else 0
```

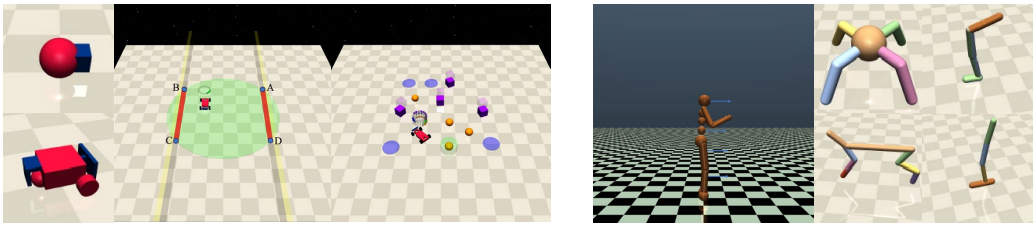
C.3 Model Rollout

During model rollout, we follow the rollout procedure used in works such as MOPO, performing a rollout after every K policy gradient updates. For each rollout, a batch of size b is sampled from $\mathcal{D}$, and each data point in the batch is rolled out for H steps. The process of sampling and branch rollout is repeated E times. In practice, we set $K = 250000$, $b = 50000$, $H = 1$, and $E = 10$.

C.4 Hyperparameters

The training of PROCO involves the selection of hyperparameters. To ensure reproducibility, this section outlines the specific hyperparameters used in our experiments, as shown in Table 2. Specifically, for the model rollout frequency, we set it to $2.5e4$ for LSPC and CAPS, and to $2.5e5$ for FISOR. PROCO is implemented based on OSRL [1], FISOR [2], LSPC [3], and CAPS [4] code bases, and

[1] https://github.com/liuzuxin/OSRL
[2] https://github.com/ZhengYinan-AIR/FISOR
[3] https://github.com/PrajwalKoirala/LSPC-Safe-Offline-RL
[4] https://github.com/yassineCh/CAPS



(a) Navigation Tasks (b) Velocity Tasks

Figure 6: Tasks used in this paper. (a) Navigation Tasks based on Point and Car robots. (b) Velocity Tasks based on Ant, HalfCheetah, Hopper, Walker2d, and Swimmer robots.

the default parameters are retained except for FISOR, we set the expectile τ to 0.9 in Navigation tasks, and to 0.95 in the Velocity tasks. Also, we set the model training step to $2e6$ in FISOR.

Table 2: Hyperparameter choices of PROCO.

	Hyperparameter	Value
model learning	model hidden layers	$[256, 256, 256, 256]$
	layer weight decays	$[2.5e-5, 5e-5, 7.5e-5, 7.5e-5, 1e-4]$
	model ensemble number	7
	model elites number	5
	batch size	512
	learning rate	0.001
cost function generation	p_{min}	10%
	p_{max}	30%
	max LLM query number	10
policy learning	model rollout frequency	$\{2.5e4, 2.5e5\}$
	model rollout batch size	50000
	model rollout length	1
	model rollout epoch	10
	model rollout std	0.1

D Detailed Description of the Tasks and Baselines

D.1 Tasks

All tasks used in this paper are derived from the Safety-Gymnasium's Navigation Tasks and Velocity Tasks. In the Navigation Tasks, there are two different types of robots: Point and Car, which we need to control to navigate through the environment and earn rewards by reaching target points, pressing the correct buttons, or moving in designated directions. Different tasks also have varying costs, such as avoiding collisions with specific targets, preventing incorrect button presses, and staying within designated boundaries.

The Velocity Tasks are built on traditional MuJoCo (Todorov et al., 2012) simulations, requiring robots such as Ant, HalfCheetah, Swimmer, and Walker2d to move, where higher speeds result in higher rewards. However, each robot has specific safety velocity thresholds for different tasks, and exceeding these thresholds leads to unsafe states. For detailed descriptions of each task, refer to the original Safety-Gymnasium paper (Ji et al., 2023).

D.2 Baselines

We provide a more detailed introduction to the baselines in this section.

- **CDT** is the previous advanced algorithm under sequence modeling for offline safe RL. After incorporating CTG into DT, CDT seeks to address the conflict between safety constraints and reward maximization. To tackle this issue, CDT proposes a data augmentation approach, where reward returns for certain safe but low-reward trajectories in the offline dataset are re-labeled with higher values.
- **CCAC** is one of the advanced algorithms for soft-constraint offline safe RL. It employs conservative estimation to obtain pessimistic value estimates and incorporates the constraint condition into the policy input, enabling the policy to handle multiple constraint thresholds simultaneously.
- **FISOR** is one of the advanced offline safe RL algorithm based on hard-constraint modeling. Similar to other algorithms under hard-constraints (Yu et al., 2022; Ganai et al., 2023), it first divides the state space into feasible and infeasible regions. It then uses the IQL (Kostrikov et al., 2022) algorithm to learn feasible value functions for offline feasible region identification. Next, FISOR sets distinct learning objectives for the feasible and infeasible regions. In the feasible region, it aims to maximize the reward while ensuring feasibility, while in the infeasible region, it focuses on minimizing constraint violations. Finally, FISOR employs a diffusion model to represent and learn the policy.
- **LSPC**-O, similar to FISOR, performs hard-constraint safe policy learning. It first trains a CVAE using the feasibility advantage as a weighting factor, such that the CVAE generates only safe actions. Subsequently, optimization is conducted in the latent space of the CVAE with the objective of maximizing reward, aiming to achieve a balance between reward optimization and safety preservation.
- **CAPS**(IQL) is also an advanced algorithm under the soft-constraint setting. Unlike methods that explicitly incorporate constraint thresholds during training, CAPS does not use the constraint threshold in its training process. Instead, similar to FISOR, it adopts IQL-style expectile regression for value function learning and employs weighted behavior cloning for policy learning. To enable decision-making under different constraint thresholds at deployment time, CAPS utilizes a multi-head policy that outputs multiple candidate actions, from which an action satisfying the specified constraint threshold and achieving high reward is selected.

E More Experimental Results

E.1 Time Complexity Analysis

To better understand the practical utility of PROCO, we evaluate its training time overhead in this section. Using the CarButton environment as an example, we report the training time measured on a single NVIDIA 4090 GPU under an otherwise idle setting (i.e., with no additional CPU/GPU workload), as summarized in Table 3.

Table 3: Time complexity analysis.

Dynamics Learning	Maximum LLM Generation (reflection for full 10 rounds)	Total Dynamics Rollout (including rollout generation and labeling over all 10 rollout epochs)	Policy Training
0.58h	0.20h	0.03h	1.88h

E.2 Sensitivity on LLM Reflection Rounds

In this section, we investigate the impact of different maximum LLM reflection rounds on policy performance. The results are presented in Table 4. As shown, increasing the maximum number of LLM reflection rounds leads to a notable improvement in the safety performance of the policy, further highlighting the importance of the reflection mechanism in enhancing the reliability of LLMs.

Table 4: Sensitive analysis on different LLM reflection rounds.

Task	0		1		5		10	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.06	3.40	0.06	2.87	0.02	1.46	0.01	1.15
PointButton2	0.07	4.55	0.07	3.39	0.05	2.11	0.08	2.76
PointGoal1	**0.34**	**0.86**	0.50	1.68	0.43	1.35	**0.35**	**0.96**
PointGoal2	**0.08**	**0.42**	**0.08**	**0.52**	**0.09**	**0.61**	**0.08**	**0.38**
PointPush1	0.20	2.17	**0.20**	**0.53**	**0.14**	**0.53**	**0.17**	**0.86**
PointPush2	0.07	1.22	0.12	1.47	0.07	1.48	**0.12**	**0.30**
CarButton1	-0.01	2.57	-0.01	2.54	**-0.05**	**0.64**	**-0.03**	**0.60**
CarButton2	0.00	3.21	-0.01	3.94	-0.03	1.26	**0.00**	**0.99**
CarGoal1	**0.16**	**0.16**	**0.12**	**0.03**	**0.15**	**0.11**	**0.13**	**0.02**
CarGoal2	**0.04**	**0.80**	**0.06**	**0.36**	**0.02**	**0.37**	**0.03**	**0.52**
CarPush1	0.11	1.86	0.16	1.41	**0.16**	**0.81**	**0.16**	**0.93**
CarPush2	0.03	1.19	0.05	2.64	0.05	3.77	**0.01**	**0.04**
SwimmerVelocityV1	**0.00**	**0.23**	**0.02**	**0.00**	**0.04**	**0.00**	**0.02**	**0.12**
HopperVelocityV1	**0.13**	**0.00**	**0.09**	**0.05**	**0.25**	**0.00**	**0.11**	**0.07**
HalfCheetahVelocityV1	**0.65**	**0.00**	**0.41**	**0.00**	**0.46**	**0.00**	**0.48**	**0.00**
Walker2dVelocityV1	0.18	2.09	0.17	1.57	**0.13**	**0.83**	**0.09**	**0.58**
AntVelocityV1	**0.50**	**0.00**	**0.55**	**0.00**	**0.45**	**0.00**	**0.52**	**0.00**
Average	0.15	1.45	0.16	1.35	**0.14**	**0.90**	**0.14**	**0.60**

E.3 Discussion about Prompt Wording

Prompt wording is indeed a key factor influencing the effectiveness of LLMs. Therefore, in this section, we provide a detailed discussion on prompt design. Our LLM prompt consists of three main components (excluding reflection): the task prompt, the cost prompt, and the instruction prompt. Since the task prompt and cost prompt contain essential information, they are difficult to modify or omit. Consequently, the primary design focus lies in the instruction prompt.

Within the instruction prompt, in addition to specifying the requirement to generate cost signals, the design mainly involves two elements: a conservative prompt ("you should be a little more conservative, situations that do not violate safety constraints but are close to doing so can also be considered unsafe") and a role-playing prompt ("you are an expert Python coder"). We have already analyzed the effect of the conservative prompt in the ablation study. Here, we further introduce a new baseline, *W/o Role*, to evaluate the necessity of the role-playing prompt. As shown in Table 5, the specially designed conservative prompt, the same as in ablation study, has a noticeable impact on the safety performance of the policy. In contrast, the role-playing prompt leads to only marginal performance differences, indicating that, when the LLM is sufficiently capable, our method is relatively robust to non-informational prompt variations.

E.4 Sensitivity on Unsafe Data Selection

In the main paper, we evaluate the performance of PROCO and FISOR under varying amounts of unsafe data. Since the selection of unsafe data is randomized, we further investigate the impact of different unsafe data selections in this section. Specifically, under the 10% unsafe data setting, we generate two additional datasets using different random seeds. Together with the dataset used in the main paper, resulting in three distinct datasets for evaluation. As shown in Table 6, PROCO is largely insensitive to the specific selection of unsafe data. This is likely because our method can generate a substantial number of unsafe samples, thereby reducing the influence of the initial unsafe dataset. In contrast, FISOR exhibits higher sensitivity due to the limited amount of unsafe data available. While different unsafe subsets may introduce some variation in safety performance on individual environments, the overall average performance-and, importantly, the issue of severe safety violations-remains largely unchanged.

Table 5: Sensitive analysis on prompt wording.

Task	PROCO		W/o Consv.		W/o Role	
	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.01	1.15	0.09	4.01	0.03	1.54
PointButton2	0.08	2.76	0.08	3.54	0.06	2.01
PointGoal1	**0.35**	**0.96**	**0.37**	**0.63**	0.42	1.35
PointGoal2	**0.08**	**0.38**	**0.06**	**0.74**	**0.06**	**0.93**
PointPush1	**0.17**	**0.86**	**0.22**	**0.94**	**0.18**	**0.45**
PointPush2	**0.12**	**0.30**	0.11	3.98	0.19	1.79
CarButton1	**-0.03**	**0.60**	-0.04	2.76	**-0.01**	**0.53**
CarButton2	**0.00**	**0.99**	-0.01	2.40	**-0.03**	**0.99**
CarGoal1	**0.13**	**0.02**	**0.23**	**0.50**	**0.11**	**0.09**
CarGoal2	**0.03**	**0.52**	**0.03**	**0.39**	**0.04**	**0.14**
CarPush1	**0.16**	**0.93**	**0.22**	**0.45**	**0.13**	**0.78**
CarPush2	**0.01**	**0.04**	0.04	3.00	**0.02**	**0.31**
SwimmerVelocityV1	**0.02**	**0.12**	**0.01**	**0.00**	**0.03**	**0.00**
HopperVelocityV1	**0.11**	**0.07**	**0.06**	**0.00**	**0.18**	**0.02**
HalfCheetahVelocityV1	**0.48**	**0.00**	**0.81**	**0.00**	**0.48**	**0.00**
Walker2dVelocityV1	**0.09**	**0.58**	0.22	2.28	**0.13**	**0.66**
AntVelocityV1	**0.52**	**0.00**	**0.41**	**0.00**	**0.50**	**0.00**
Average	**0.14**	**0.60**	0.17	1.51	**0.15**	**0.68**

Table 6: Sensitive analysis on different unsafe data.

Task	Ours Rand1		FISOR Rand1		Ours Rand2		FISOR Rand2		Ours Rand3		FISOR Rand3	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.05	1.33	0.40	9.75	0.03	1.55	0.41	9.11	0.02	1.48	0.45	11.24
PointButton2	0.05	2.24	0.44	10.78	0.05	2.34	0.42	12.15	0.07	1.52	0.42	11.82
PointGoal1	0.44	1.63	0.70	4.24	0.41	1.15	0.72	4.80	0.43	1.90	0.70	4.77
PointGoal2	**0.07**	**0.83**	0.56	11.04	**0.08**	**0.69**	0.60	10.76	**0.05**	**0.87**	0.60	10.62
PointPush1	**0.18**	**0.77**	0.35	3.52	0.16	1.11	0.39	2.81	**0.18**	**0.74**	0.38	3.17
PointPush2	0.14	1.40	0.28	6.17	0.16	1.60	0.26	5.30	0.13	1.36	0.28	5.73
CarButton1	**-0.06**	**0.51**	0.35	15.25	**-0.01**	**0.79**	0.37	15.64	-0.02	1.03	0.35	20.97
CarButton2	-0.02	1.21	0.27	19.57	-0.03	1.41	0.26	16.21	-0.03	1.31	0.24	16.46
CarGoal1	**0.15**	**0.02**	0.72	3.89	**0.13**	**0.00**	0.65	4.53	**0.13**	**0.00**	0.66	4.42
CarGoal2	**0.03**	**0.14**	0.45	10.80	**0.01**	**0.46**	0.42	11.14	**0.04**	**0.16**	0.51	10.67
CarPush1	**0.17**	**0.46**	0.43	1.89	**0.16**	**0.69**	0.42	2.93	**0.16**	**0.86**	0.34	2.22
CarPush2	**0.03**	**0.69**	0.34	7.55	**0.05**	**0.04**	0.33	5.95	**0.04**	**0.83**	0.31	6.60
SwimmerVelocityV1	**0.04**	**0.00**	-0.02	6.10	**0.05**	**0.00**	-0.05	7.84	**0.04**	**0.01**	-0.05	4.02
HopperVelocityV1	**0.10**	**0.01**	0.16	1.51	**0.11**	**0.00**	0.11	2.78	**0.05**	**0.00**	0.08	3.21
HalfCheetahVelocityV1	**0.42**	**0.00**	**0.92**	**0.00**	**0.43**	**0.00**	**0.95**	**0.00**	**0.40**	**0.00**	**0.92**	**0.02**
Walker2dVelocityV1	0.12	1.15	0.52	4.14	**0.13**	**0.97**	0.10	1.59	**0.09**	**0.89**	0.15	2.29
AntVelocityV1	**0.49**	**0.00**	**0.86**	**0.94**	**0.48**	**0.00**	0.91	1.13	**0.45**	**0.00**	0.94	1.62
Average	**0.14**	**0.73**	0.46	6.89	**0.14**	**0.75**	0.43	6.75	**0.13**	**0.76**	0.43	7.05

E.5 Integrate PROCO with Soft-constraint Algorithms

In this section, we further integrate PROCO with three soft-constraint offline safe reinforcement learning algorithms-**CPQ** (Xu et al., 2022), **BCQ Lagrange (BCQ-L)** (Fujimoto et al., 2019), and **COptiDICE** (Lee et al., 2022)-and evaluate the impact of the model-based rollout data generated by PROCO on these methods. The results are shown in Table 7. We observe that all three algorithms benefit from the rollout data provided by PROCO, exhibiting clear improvements in safety performance, which validates the effectiveness of the proposed model-based rollout mechanism.

Nevertheless, even after these improvements, their safety performance remains relatively unsatisfactory, further indicating that feasibility-based hard-constraint methods are more suitable for this class of scenarios.

Table 7: Results of integrate PROCO with soft-constraint algorithms.

Task	CPQ		BCQ-L		COptiDICE		CPQ PROCO		BCQ-L PROCO		COptiDICE PROCO	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.77	13.37	0.25	4.64	0.06	2.80	0.23	7.14	0.07	2.44	0.14	2.92
PointButton2	0.62	14.53	0.41	8.65	0.14	4.86	0.39	12.07	0.14	2.83	0.16	4.42
PointGoal1	0.80	5.39	0.71	3.89	0.50	5.34	0.30	4.03	0.82	3.28	0.52	4.65
PointGoal2	0.80	18.69	0.66	11.57	0.38	5.15	0.09	8.95	0.26	4.00	0.34	6.50
PointPush1	0.19	6.35	0.31	2.76	0.09	3.58	0.14	2.13	0.24	1.72	0.12	2.50
PointPush2	0.22	10.21	0.24	4.03	0.03	3.00	0.08	2.94	0.12	2.86	-0.01	3.34
CarButton1	0.35	42.38	-0.02	5.56	-0.04	3.69	0.06	7.08	-0.10	2.42	-0.14	1.85
CarButton2	0.54	34.06	0.00	5.15	-0.08	2.93	0.09	10.62	-0.10	3.73	-0.15	3.01
CarGoal1	0.80	5.74	0.47	2.86	0.38	2.14	0.46	3.04	0.35	2.36	0.31	1.73
CarGoal2	0.82	20.41	0.25	3.67	0.30	3.72	0.27	6.12	0.25	3.30	0.16	1.59
CarPush1	-0.01	5.08	0.20	1.59	0.20	2.21	-0.13	1.76	0.19	1.02	0.22	1.76
CarPush2	0.29	16.36	0.15	6.94	0.07	3.00	0.14	2.73	0.11	3.40	0.11	2.95
SwimmerVelocityV1	0.06	11.65	0.56	24.28	0.48	3.43	**-0.03**	**0.05**	0.39	3.20	0.43	4.35
HopperVelocityV1	0.06	5.09	0.36	7.66	0.18	4.59	0.07	4.57	0.40	5.77	0.38	2.73
HalfCheetahVelocityV1	1.35	88.45	1.01	25.83	**0.59**	**0.00**	**0.18**	**0.01**	0.91	8.95	**0.58**	**0.00**
Walker2dVelocityV1	0.03	1.08	**0.79**	**0.24**	0.10	1.68	**0.02**	**0.32**	**0.75**	**0.99**	0.11	2.06
AntVelocityV1	**-1.01**	**0.00**	0.92	27.89	0.98	5.16	**-0.92**	**0.00**	**0.84**	**0.11**	0.97	3.86
Average	0.39	17.58	0.43	8.66	0.26	3.37	0.09	4.33	0.33	3.08	0.25	2.95
Safe Percent	1/17		1/17		1/17		4/17		2/17		1/17	
Safety Improvement	/		/		/		1325%		558%		42%	

E.6 Standard Deviation of the Main Results

In this section, we present the standard deviation of PROCO and various baselines computed with three different random seeds, as shown in Table 8. The results demonstrate that variants of PROCO achieve lower standard deviation compared to other baselines, indicating their stability.

Table 8: Detailed standard deviation results of the main experiment. All results are computed using three different random seeds.

Task	BC		CDT		CCAC		LSPC		CAPS		FISOR		LSPC PROCO		CAPS PROCO		FISOR PROCO	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.05	1.21	0.14	1.90	0.00	1.76	0.01	0.86	0.11	1.87	0.12	0.91	0.04	0.32	0.02	0.34	0.01	0.56
PointButton2	0.08	1.57	0.02	2.28	0.06	0.89	0.13	1.39	0.04	0.79	0.06	0.41	0.08	2.30	0.02	0.18	0.02	0.88
PointGoal1	0.03	0.13	0.01	0.55	0.00	1.03	0.04	1.04	0.05	0.61	0.01	0.52	0.07	0.10	0.05	0.00	0.00	0.44
PointGoal2	0.05	2.15	0.01	0.60	0.02	4.01	0.04	0.43	0.02	1.42	0.03	1.01	0.03	0.27	0.03	0.41	0.05	0.43
PointPush1	0.03	0.91	0.03	0.15	0.04	1.66	0.01	1.11	0.03	0.84	0.06	1.27	0.01	1.38	0.02	0.33	0.00	0.53
PointPush2	0.06	1.04	0.02	0.89	0.09	2.46	0.02	3.53	0.02	0.92	0.06	1.79	0.04	0.62	0.03	1.82	0.02	0.18
CarButton1	0.01	1.35	0.02	1.86	0.05	24.41	0.07	1.53	0.08	3.29	0.03	1.78	0.02	2.43	0.01	1.30	0.04	0.23
CarButton2	0.10	1.18	0.03	1.15	0.40	8.62	0.00	1.13	0.06	1.46	0.04	0.94	0.07	0.63	0.04	0.23	0.00	0.21
CarGoal1	0.01	0.56	0.04	0.15	0.00	0.00	0.08	0.41	0.06	0.55	0.01	0.91	0.06	0.31	0.11	0.10	0.02	0.03
CarGoal2	0.03	0.38	0.01	0.38	0.34	10.96	0.06	0.51	0.08	1.77	0.02	0.20	0.06	0.34	0.03	0.26	0.02	0.50
CarPush1	0.05	0.24	0.01	1.13	0.33	0.17	0.01	0.53	0.01	0.82	0.04	0.35	0.02	0.74	0.03	0.09	0.02	0.86
CarPush2	0.03	1.21	0.01	1.43	0.08	4.27	0.02	0.55	0.02	1.67	0.06	1.84	0.01	0.63	0.03	0.57	0.00	0.06
SwimmerVelocityV1	0.10	5.05	0.02	3.13	0.05	3.57	0.04	4.55	0.13	6.07	0.01	0.56	0.06	0.59	0.03	0.12	0.02	0.12
HopperVelocityV1	0.17	0.97	0.03	0.66	0.05	3.82	0.05	1.04	0.13	2.47	0.01	0.15	0.03	4.20	0.03	0.04	0.01	0.10
HalfCheetahVelocityV1	0.00	2.94	0.01	1.01	0.07	0.65	0.00	3.84	0.01	2.85	0.01	3.45	0.05	0.00	0.01	0.00	0.01	0.00
Walker2dVelocityV1	0.09	1.17	0.02	0.60	0.03	1.13	0.01	0.00	0.02	0.53	0.03	0.95	0.08	1.31	0.09	0.78	0.03	0.09
AntVelocityV1	0.01	3.91	0.00	0.31	0.00	0.00	0.00	1.48	0.00	0.31	0.00	0.76	0.01	0.21	0.06	0.00	0.04	0.00
Average	0.05	1.53	0.03	1.07	0.09	4.08	0.03	1.41	0.05	1.66	0.04	1.05	0.04	0.96	0.04	0.39	0.02	0.31

E.7 Detailed ablation results

In this section, we also provide the detailed results of the ablation studies, as shown in Table 9.

Table 9: Detailed ablation results (mean±std).

Task	W/o Model		Full Model		W/o Relabel	
	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.06±0.00	2.84±0.58	0.01±0.00	1.82±0.76	0.07±0.01	1.10±0.39
PointButton2	0.09±0.03	4.32±0.28	0.03±0.02	2.14±0.72	0.09±0.02	2.19±0.36
PointGoal1	0.54±0.04	2.41±0.76	0.07±0.02	0.18±0.13	0.28±0.04	0.25±0.13
PointGoal2	0.14±0.03	1.58±0.68	0.03±0.01	0.63±0.34	0.09±0.05	0.92±0.30
PointPush1	0.25±0.02	1.03±0.32	0.13±0.04	0.79±0.68	0.21±0.02	0.60±0.32
PointPush2	0.11±0.03	1.21±0.36	0.04±0.05	0.48±0.25	0.07±0.03	1.22±0.27
CarButton1	-0.02±0.01	1.24±0.25	0.00±0.00	1.57±1.22	-0.03±0.03	1.22±0.37
CarButton2	0.00±0.02	3.22±0.94	-0.06±0.01	1.99±0.80	-0.03±0.03	1.76±1.04
CarGoal1	0.27±0.02	0.37±0.13	0.02±0.02	0.00±0.00	0.15±0.06	0.31±0.18
CarGoal2	0.06±0.03	0.14±0.12	-0.01±0.02	0.07±0.07	0.11±0.02	1.00±0.34
CarPush1	0.24±0.04	1.03±0.69	0.16±0.03	0.12±0.17	0.17±0.02	0.29±0.19
CarPush2	0.03±0.03	0.48±0.03	-0.03±0.05	0.00±0.00	0.06±0.02	1.27±0.53
SwimmerVelocityV1	0.01±0.01	0.00±0.00	0.02±0.00	0.00±0.00	0.01±0.02	0.07±0.10
HopperVelocityV1	0.17±0.05	0.17±0.12	0.04±0.03	0.00±0.00	0.12±0.09	0.18±0.20
HalfCheetahVelocityV1	0.51±0.04	0.00±0.00	0.23±0.05	0.00±0.00	0.83±0.04	1.40±1.89
Walker2dVelocityV1	0.15±0.03	1.47±0.50	0.05±0.03	0.01±0.01	0.18±0.05	3.36±1.38
AntVelocityV1	0.59±0.01	0.00±0.00	0.00±0.08	0.01±0.01	0.52±0.05	0.01±0.01
Average	0.19	1.27	0.05	0.58	0.17	1.01

Task	Det. Rollout		W/o Consv.		W/o Refl.	
	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.02±0.04	1.67±0.97	0.09±0.03	4.01±0.27	0.06±0.03	3.40±1.70
PointButton2	0.07±0.01	2.90±0.64	0.08±0.01	3.54±0.80	0.07±0.01	4.55±1.48
PointGoal1	0.40±0.02	1.65±0.36	0.37±0.00	0.63±0.21	0.34±0.04	0.86±0.55
PointGoal2	0.06±0.01	0.25±0.18	0.06±0.01	0.74±0.54	0.08±0.02	0.42±0.13
PointPush1	0.18±0.04	0.91±1.10	0.22±0.02	0.94±0.14	0.20±0.05	2.17±2.38
PointPush2	0.08±0.04	0.51±0.38	0.11±0.04	3.98±1.95	0.07±0.01	1.22±0.90
CarButton1	-0.04±0.04	1.75±0.59	-0.04±0.02	2.76±1.10	-0.01±0.03	2.57±1.32
CarButton2	-0.01±0.01	1.36±0.27	-0.01±0.00	2.40±0.72	0.00±0.01	3.21±1.44
CarGoal1	0.15±0.04	0.03±0.04	0.23±0.04	0.50±0.37	0.16±0.02	0.16±0.14
CarGoal2	0.03±0.00	0.08±0.11	0.03±0.00	0.39±0.49	0.04±0.03	0.80±0.93
CarPush1	0.18±0.03	1.87±1.23	0.22±0.03	0.45±0.24	0.11±0.04	1.86±2.43
CarPush2	0.00±0.00	0.04±0.05	0.04±0.03	3.00±1.34	0.03±0.04	1.19±0.97
SwimmerVelocityV1	0.00±0.01	0.03±0.02	0.01±0.01	0.00±0.00	0.00±0.00	0.23±0.33
HopperVelocityV1	0.20±0.12	0.01±0.01	0.06±0.08	0.00±0.00	0.13±0.17	0.00±0.00
HalfCheetahVelocityV1	0.49±0.02	0.00±0.00	0.81±0.01	0.00±0.00	0.65±0.07	0.00±0.00
Walker2dVelocityV1	0.15±0.01	1.57±0.36	0.22±0.04	2.28±0.15	0.18±0.00	2.09±0.57
AntVelocityV1	0.46±0.05	0.00±0.00	0.41±0.03	0.00±0.00	0.50±0.18	0.00±0.00
Average	0.14	0.86	0.17	1.51	0.15	1.45

E.8 Detailed Sensitivity Analysis Results

Finally, in this section, we provide the detailed results of sensitivity studies, as shown in Table 10. Notably, while safety performance degrades significantly when $H = 5$, this degradation is mainly observed in the Point and Car tasks, whereas the MuJoCo Velocity tasks do not exhibit such a decline and may even become more conservative. These results suggest that when the environment model is sufficiently accurate, increasing H can indeed yield improved safety performance; however, if the environment model accuracy is low, keeping H small is preferable to avoid the negative impact of model errors on learning stability.

F Impact Statement

The goal of this work is to advance offline safe RL under safe-only datasets, thereby enabling the deployment of learned policies in a broader range of real-world applications. For example, in embodied

Table 10: Detailed sensitivity analysis results (mean±std).

Task	$E = 2$		$E = 5$		$H = 2$	
	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.03±0.01	1.93±0.14	0.03±0.00	1.55±0.55	-0.02±0.01	0.57±0.20
PointButton2	0.06±0.04	2.31±1.15	0.06±0.01	2.42±0.58	-0.06±0.05	1.79±0.33
PointGoal1	0.45±0.03	1.70±1.03	0.42±0.07	1.28±0.66	0.14±0.03	2.67±0.31
PointGoal2	0.10±0.04	0.71±0.45	0.08±0.02	0.53±0.31	-0.04±0.04	1.20±0.27
PointPush1	0.20±0.02	0.43±1.35	0.18±0.02	1.14±0.53	0.13±0.05	1.10±0.44
PointPush2	0.10±0.00	0.76±0.31	0.09±0.02	0.24±0.16	0.06±0.02	0.65±0.25
CarButton1	-0.04±0.05	0.61±0.42	-0.07±0.05	0.31±0.11	-0.04±0.03	0.94±0.04
CarButton2	-0.02±0.04	1.85±0.29	-0.03±0.03	1.68±0.43	-0.13±0.06	0.75±0.27
CarGoal1	0.22±0.02	0.15±0.10	0.20±0.02	0.03±0.05	0.16±0.08	1.45±0.59
CarGoal2	0.05±0.02	0.58±0.26	0.05±0.00	0.77±0.11	0.05±0.04	2.29±0.89
CarPush1	0.19±0.04	0.35±0.22	0.17±0.03	0.45±0.39	0.12±0.02	0.32±0.09
CarPush2	0.03±0.01	0.01±0.01	-0.01±0.02	0.00±0.00	-0.01±0.04	0.64±0.91
SwimmerVelocityV1	0.01±0.00	0.11±0.15	0.02±0.01	0.32±0.45	0.01±0.01	0.00±0.00
HopperVelocityV1	0.18±0.02	0.21±0.14	0.38±0.15	0.03±0.03	0.13±0.04	0.14±0.19
HalfCheetahVelocityV1	0.50±0.04	0.00±0.00	0.49±0.03	0.00±0.00	0.39±0.07	0.00±0.00
Walker2dVelocityV1	0.13±0.03	1.50±0.40	0.11±0.04	1.26±0.59	0.05±0.03	0.25±0.09
AntVelocityV1	0.59±0.00	0.00±0.00	0.53±0.01	0.00±0.00	0.29±0.05	0.00±0.00
Average	0.16	0.77	0.16	0.70	0.07	0.87

Task	$H = 5$		Deepseek R1		Gemini 2.5 Pro	
	r↑	c↓	r↑	c↓	r↑	c↓
PointButton1	0.21±0.13	10.36±1.13	0.05±0.01	2.31±0.57	0.01±0.02	1.18±0.64
PointButton2	0.27±0.07	11.76±2.04	0.08±0.03	2.57±0.60	0.07±0.02	1.80±0.27
PointGoal1	0.06±0.13	5.19±3.41	0.35±0.03	0.71±0.24	0.31±0.02	0.63±0.20
PointGoal2	0.26±0.06	6.49±3.44	0.05±0.03	0.11±0.08	0.10±0.04	0.68±0.31
PointPush1	0.11±0.02	1.04±0.99	0.25±0.07	1.79±1.42	0.17±0.02	1.04±0.39
PointPush2	0.10±0.02	1.69±2.18	0.12±0.02	0.82±0.48	0.10±0.01	0.51±0.35
CarButton1	-0.01±0.02	6.75±8.33	-0.03±0.04	1.04±0.24	-0.03±0.01	0.60±0.24
CarButton2	0.03±0.01	13.28±4.62	0.00±0.01	1.94±0.34	-0.03±0.02	1.02±0.21
CarGoal1	0.16±0.09	3.11±2.03	0.15±0.04	0.11±0.06	0.11±0.00	0.03±0.04
CarGoal2	0.06±0.04	8.77±7.05	0.02±0.02	0.60±0.33	0.03±0.01	0.38±0.42
CarPush1	0.18±0.00	1.57±1.23	0.22±0.07	0.85±0.47	0.13±0.02	0.49±0.08
CarPush2	0.06±0.01	4.16±1.62	0.07±0.05	0.42±0.06	0.04±0.02	0.15±0.21
SwimmerVelocityV1	0.04±0.02	0.00±0.00	-0.01±0.01	0.00±0.00	0.00±0.03	0.00±0.00
HopperVelocityV1	0.06±0.05	0.47±0.35	0.29±0.10	1.27±1.39	0.20±0.03	0.25±0.27
HalfCheetahVelocityV1	0.42±0.07	0.00±0.00	0.08±0.02	0.00±0.00	0.54±0.01	0.00±0.00
Walker2dVelocityV1	0.12±0.04	0.43±0.26	0.17±0.02	2.03±0.46	0.15±0.02	1.66±0.61
AntVelocityV1	0.49±0.01	0.00±0.00	0.54±0.11	0.00±0.00	0.39±0.10	0.00±0.00
Average	0.15	4.41	0.14	0.97	0.14	0.61

intelligence, data are often collected via human teleoperation; when an imminent unsafe collision is detected, data collection may be interrupted, or external interventions may alter the agent's state or the environment to prevent collisions. The resulting datasets naturally contain only safe trajectories, yet they are highly valuable for training safe policies. Our method is specifically designed to operate under such realistic data collection paradigms. Furthermore, the work presented in this paper does not introduce any additional ethical concerns beyond those commonly associated with RL research. Therefore, no special discussion of ethical issues is required.

G Discussions and Limitations

In this work, although LLMs are employed as the primary approach for cost function generation, our core objective is not to rely on LLMs per se. Rather, we view LLMs as a knowledge processing tool that transforms natural-language safety specifications into a cost function-an intermediate representation that can be effectively aligned with data. Consequently, alternative approaches that serve a similar role can also be naturally used. For instance, human experts could directly process and translate safety descriptions into a cost function, which can then be iteratively refined based on feedback from the data. Compared to manual efforts, the use of LLMs offers a higher degree of automation and reduced human labor, while a potential drawback lies in the additional computational and monetary costs associated with LLM inference.

Meanwhile, although this work effectively addresses the challenge of learning offline safe policies in scenarios with scarce or no unsafe data, some limitations remain. The first lies in the accuracy and generalization of the environment model. When model accuracy is low, the rollout horizon must be restricted. A potential solution is to employ more powerful generative models, such as diffusion models, as environment models. The second limitation concerns the generation of conservative cost functions: the cost description must be grounded in features observable in the agent's observations. If certain variables required to compute the cost function cannot be directly or indirectly inferred from the observations, our approach cannot be applied. Additionally, despite the adoption of a check-and-feedback mechanism, the reliability of LLM outputs remains affected by hallucinations. A promising direction for future research is to integrate human-in-the-loop strategies to mitigate this issue. Finally, due to the reliance on cost function generation, the current PROCO is restricted to state-based tasks and cannot be applied to vision-based tasks. This is because VLMs cannot directly produce sufficiently accurate image-based cost functions, and performing VLM-based evaluation for every sample would be prohibitively expensive.