PREFINE: Preference-Based Implicit Reward and Cost Fine-Tuning for Safety Alignment

Anonymous Author(s)
Submission Id: 525

ABSTRACT

We address the problem of making a pre-trained reinforcement learning (RL) policy safety-aware by incorporating cost constraints without retraining it from scratch. While costs could be numerically encoded, we assume a more general setting is when costs are provided as preferences. Given a reward-optimized policy and a small dataset of preferred (low-cost) and dispreferred (high-cost) trajectories, our goal is to fine-tune the policy to generate low-cost behaviors while retaining high rewards. Unlike standard RLHF in language models, where preferences are defined over responses to the same prompt, our setting involves trajectory-level preferences in continuous control environments. We introduce **PREFINE**: **Pre**ference-based Implicit Reward and Cost **Fine**-Tuning for Safety Alignment which is a preference-based fine-tuning method that adapts **Direct Preference Optimization (DPO)**, which is now widely used for LLM fine-tuning, to the sequential decision making setting. PREFINE constructs policy-sampled counterfactual trajectories to establish meaningful preference contrasts and jointly optimizes for reward retention and safety alignment. Empirically, PREFINE reduces constraint violations and catastrophic failures by over 60% while maintaining original reward behavior. PREFINE produces policies that achieve low-cost, high-reward performance with significantly improved data and computational efficiency compared to full offline RL or imitation learning, bridging preference alignment and safe policy adaptation in continuous domains.

KEYWORDS

Policy adaptation, Preference-based Optimization, Safe RL

ACM Reference Format:
Anonymous Author(s). 2026. PREFINE: Preference-Based Implicit Reward and Cost Fine-Tuning for Safety Alignment. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 14 pages.

1 INTRODUCTION

As the focus in machine learning and AI shifts from prediction to prescription, safety of AI applications is now of paramount importance. In RL [19], safety constraints are modeled by extending the MDP with cost constraints and using Lagrangian methods [1, 10, 18, 21] to solve the resulting constrained optimization problem. However, this approach is not computationally feasible as it involves solving a nested optimization problem. Likewise, standard imitation learning (IL) [3, 16, 17] focuses solely on replicating expert demonstrations and disregards non-preferred, unsafe trajectories, leaving no mechanism to enforce safety constraints. Instead, there is a growing demand for solutions that can leverage offline data and fine-tune a policy to meet safety constraints. Inspired by the work in RLHF in LLMs [5], we propose using an extension of Direct Preference Optimization (DPO) to incorporate safety constraints while ensuring that the *safety alignment* is fully offline. We introduce **PREFINE**: **Pre**ference-based Implicit Reward **Fine**-Tuning for Safety Alignment, a fully offline framework that transforms safety alignment into a preference-based learning problem. Rather than requiring hand-crafted cost signals or reward information, PREFINE relies on pairwise comparisons between safe (preferred) and unsafe (non-preferred) trajectories to implicitly encode safety without any online interaction. We combine Direct Preference Optimization (DPO) [15] with supervised fine-tuning (SFT) [14] to incorporate cost constraints into any differentiable pre-trained policy (RL or IL) without explicit cost estimation. DPO is widely used to fine-tune Large Language Models (LLMs) using human preferences. We adapt it to the sequential decision making setting in two steps: (i) we create a set of counterfactual trajectories using an existing pre-trained policy which share common states with the given trajectories in the preference dataset, and (ii) we use both reward and preference optimization to fine-tune the policy. Note that in a real world setting, such as autonomous driving, unsafe (or high cost) data is hard to collect. Therefore, PREFINE uses a mix of abundant safe (preferred) data and scarce unsafe (non-preferred) data to perform safety alignment. We show an overview of our approach in Figure 1. The expert policy π_{ref} learns from the high-reward region of the underlying DSRL [11] dataset (see Figure 1(Top-left). PREFINE leverages mixed-quality datasets spanning the reward-cost Pareto frontier. Safe (i.e. preferred) demonstrations are marked in green while the unsafe (i.e. non-preferred) demonstrations have been highlighted in red (see Figure 1(Bottom-left)). In Figure 1(Right), we see that expert policy (π_{ref}) rollouts show that it achieves high rewards but exhibits unsafe cost variations (black). PREFINE learns the policy π_{safe} and it is evident from the rollouts that π_{safe} avoids high-cost regions while retaining expert-level performance (blue).

Our hybrid DPO+SFT objective mirrors the Regularized Preference Optimization [12] principle i.e., combining a preference optimization term with an imitation (SFT) regularizer to mitigate over-optimization, formalized for RLHF settings. We adapt this structure to sequential decision-making with offline safety preferences, using SFT to anchor policy updates under partial coverage while optimizing preference-consistent comparisons. Our approach has several advantages and novel features including (i) We show that we can create policies which result in low-cost trajectories while maintaining high rewards and (ii) PREFINE is substantially more cost (time) efficient and (iii) we work in a setting that is neither imitation learning nor complete offline RL.

The main contributions of the work are as follows:

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

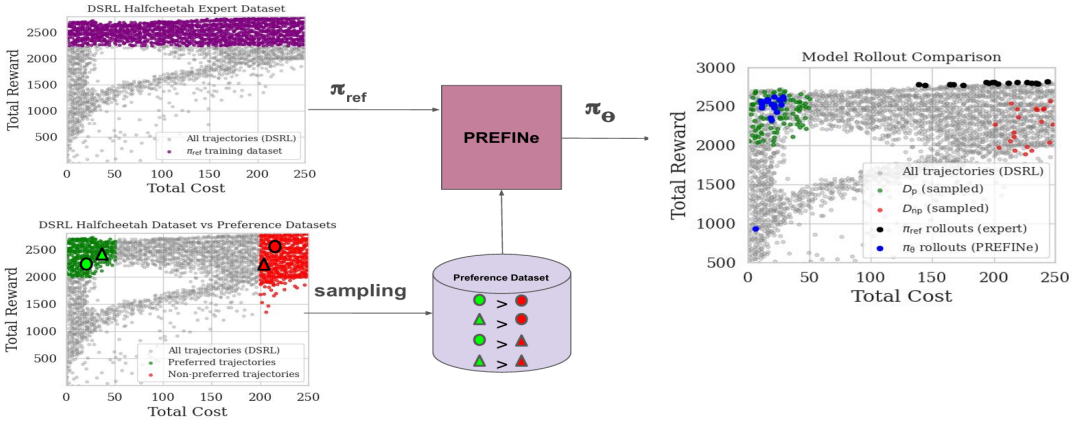

Figure 1: Overview of the PREFINE pipeline. (Top-left) The DSRL HalfCheetah offline dataset (grey) contains trajectories with a wide range of costs and rewards; we pre-train a reference policy π_{ref} on the high-reward, low-cost subset (purple). (Bottom-left) We sample a small *preferred* set $\mathcal{D}_p$ (green) of safe trajectories and a *non-preferred* set $\mathcal{D}_{np}$ (red) of unsafe trajectories to form pairwise comparisons. (Center) PREFINE ingests π_{ref} and these preference pairs, then fine-tunes in a single-stage DPO–SFT loop to produce a new policy π_θ. (Right) Rollouts of π_θ (blue) shift into the low-cost, high-reward region, retaining the performance of original π_{ref} rollouts (black) and avoiding unsafe behaviors (red) without any online interaction.

- We formulate safety alignment as a preference-based learning problem, using trajectory preferences, enabling fully offline policy updates for retrofitting cost constraints into existing RL policies.
- We introduce PREFINE, a novel approach to combine DPO's preference ranking with SFT's stability that enables direct policy learning while eliminating the need for challenging min-max optimization. We adapt DPO (for the first time) from LLMs to safety alignment in RL.
- We show that PREFINE reduces constraint violations and catastrophic failures by 60% - 92% while retaining expert-level task performance, and converges with a wall-clock time an order of magnitude smaller compared to cost-based and distribution-matching baselines.

For evaluation, we use the well-established DSRL benchmark [11], a safe offline RL evaluation benchmark. We present the results of extensive experiments that we conducted to compare our method with several state-of-the-art baselines in terms of normalized reward and normalized cost across 12 continuous control tasks in two popular domains. Note that preserving policy performance in terms of reward is prioritized over minimizing cost, which here serves as a proxy for safety. PREFINE is designed for scenarios where costs are better defined as preferences and multiple decision trajectories achieve the same goal but differ in safety, such as autonomous driving or surgical interventions. In driving, for instance, reaching a destination is the objective, yet overspeeding makes the trajectory unsafe. Similarly, a surgery may succeed overall, even if certain intermediate actions are risky. In such settings, humans can more naturally express relative safety preferences (e.g., "which driving trajectory is safer?") than assign precise numerical safety scores (e.g., "this driving trajectory rates 7.3/10"). **We show that PREFINE successfully learns policies in the setting where costs are not quantitatively defined but expressed as preferences and it retains the high rewards of the pre-trained reference policy while becoming safety-aware, outperforming the state-of-the-art baselines.** We demonstrate that PREFINE provides a fast, scalable recipe for retrofitting pre-trained policies with safety (cost constraints) using *safety alignment*, paving the way for a broader real-world deployment of RL systems. PREFINE code and datasets are available here [1].

2 PRELIMINARIES

In principle, safe RL problems are often modeled as using *Constrained Markov Decision Processes* (CMDPs) [2], defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, c, \mu_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ describes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $c : \mathcal{S} \times \mathcal{A} \to [0, C_{\max}]$ is the cost function, where $C_{\max}$ denotes the maximum allowable cost. The distribution $\mu_0 : \mathcal{S} \to [0, 1]$ represents the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. Let $\pi(a \mid s)$ denote the probability of taking action a in state s under policy π. The reward-based state-value function is defined as $V_r^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) \mid s_0 = s\right]$, and the cost-based value function as $V_c^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t c(s_t, a_t) \mid s_0 = s\right]$.

While CMDPs assume access to explicit cost signals, in many real-world safety-critical domains such costs are unavailable or hard to specify. Instead, we often have access to **pairwise preferences** between trajectories, reflecting which behaviors are safer or more desirable. Preference-based learning provides a way to incorporate such feedback.

2.1 Reinforcement Learning from Human Feedback (RLHF)

In reinforcement learning from human feedback (RLHF), the goal is to align a policy π_θ with human preferences using a dataset of pairwise comparisons $\mathcal{D} = \{(x, y_w, y_l)\}$, where x is a prompt or context, y_w is a preferred response, and y_l is a less preferred one. A common probabilistic formulation for such comparisons is

[1] https://github.com/zxXhVi/PREFINE

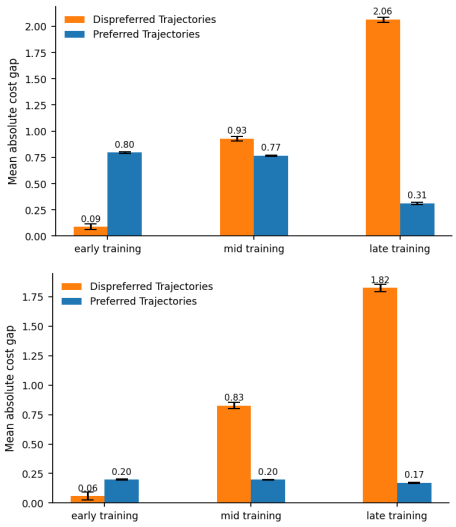

Figure 2: Mean absolute cost gap between dataset and sampled action costs across early, mid, and late training for preferred (blue) and dispreferred (orange) trajectories in HalfCheetahVelocity (top) and Walker2dVelocity (bottom). For preferred states, the gap narrows as training progresses, showing improved alignment with low-cost behaviors, while for dispreferred states, the widening gap indicates the policy increasingly avoids high-cost (unsafe) actions.

the **Bradley–Terry model** [4], which expresses the probability of preferring y_w over y_l as a softmax over their latent rewards:

$$\Pr[\, y_w \succ y_l \mid x] = \frac{\exp\big(r_E(x, y_w)\big)}{\exp\big(r_E(x, y_w)\big) + \exp\big(r_E(x, y_l)\big)}, \tag{1}$$

where $r_E(x, y)$ denotes an implicit expert reward function. This model provides a probabilistic link between preference data and latent rewards.

Direct Preference Optimization (DPO) [15] builds on this formulation by directly optimizing the policy π_θ with respect to preference pairs, without learning r_E explicitly. Given (x, y_w, y_l), DPO minimizes:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\log \sigma\Big(\beta\Big[\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\Big]\Big), \tag{2}$$

where π_{ref} is a reference policy and β controls preference sharpness. This objective implicitly defines a reward model $\hat{r}(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$, encouraging π_θ to assign higher likelihood to preferred responses while remaining close to π_{ref}.

Supervised Fine-Tuning (SFT) complements this by maximizing the likelihood of desirable responses:

$$\mathcal{L}_{\text{SFT}}(\pi_\theta) = -\mathbb{E}_{(x,y_w)\sim\mathcal{D}}\big[\log \pi_\theta(y_w|x)\big], \tag{3}$$

which stabilizes optimization by anchoring the policy on preferred data, although it does not explicitly penalize undesirable outputs.

In this work, we adapt these methods, which are originally developed for aligning large language models with human preferences, to the offline sequential decision-making setting. Bradley-Terry model provides the statistical foundation for modeling preferences, DPO introduces a principled contrastive loss against a reference policy, and SFT anchors learning to safe demonstrations. Together, these components form the basis for our preference-guided fine-tuning framework for sequential decision making.

3 PREFINE

In this section, we describe PREFINE's formulation as a preference-based fine-tuning procedure for post-hoc safety alignment in an offline sequential decision-making setting. We present a hybrid loss combining Direct Preference Optimization (DPO) [15] with supervised fine-tuning (SFT) [14], and the training algorithm.

3.1 Problem Formulation

We consider the task of *offline post-hoc safety alignment*, where the goal is not to learn a safe policy from scratch, but to *retrofit an existing policy* π_{ref} with safety (or cost constraints). The reference policy π_{ref} is assumed to be a differentiable policy trained beforehand (via RL or IL) to achieve high reward. The environment is modeled as a constrained Markov Decision Process (CMDP), defined by reward $r(s, a)$, cost $c(s, a)$, and a cost threshold τ. A trajectory is considered *safe enough* if its cumulative cost lies below τ.

Problem Definition: Given a reference policy π_{ref}, a preference data set D_p of high reward and low cost trajectories from π_{ref} and a data set D_{np} of non-preferred high reward and high cost trajectories. Let ϕ be a trajectory and let $P_\theta(\phi)$ be the probability of selecting a trajectory ϕ when using policy π_θ. Let τ be the cost threshold. The objective is to finetune π_{ref} using cost preferences in an offline manner such that the resulting π_θ satisfies the following:

$$\mathop{\mathbf{E}}_{\phi\sim P_\theta}[R(\phi)] \equiv \mathop{\mathbf{E}}_{\phi\sim P_{ref}}[R(\phi)] \tag{4}$$

$$\mathop{\mathbf{E}}_{\phi\sim P_\theta}[c(\phi)] \leq \tau \tag{5}$$

This distinguishes our setting from imitation learning (which learns directly from demonstrations) and offline RL (which learns from scratch with batch data and focuses on reward maximization). Instead, we formulate PREFINE as a *preference-based policy refinement problem*, aligning π_{ref} with safety preferences while retaining its original reward capabilities. This means that PREFINE starts from a strong policy π_{ref} and only wants to reduce the costs without sacrificing the reward. This hybrid positioning is coherent: PREFINE uses costs defined as preferences (safe vs unsafe actions) rather than explicit costs, and fine-tunes a base policy rather than re-training from scratch. **Note that the "cost" is just a proxy for the notion of safety. PREEFINe can work in settings where costs cannot be defined on a scale and are qualitative in nature for example, safe vs. rash driving by an autonomous vehicle, as explained in Section1**. We achieve this through a *hybrid preference optimization objective*, combining Direct Preference Optimization (DPO) to align with pairwise cost preferences, and a supervised fine-tuning (SFT) anchor to stabilize reward retention. The trajectory dataset $\mathcal{D}$ is partitioned into preferred (safe) trajectories D_p and non-preferred (unsafe) trajectories D_{np} based on the cost threshold and the reward. PREFINE then aligns π_{ref} with π_θ using pairwise cost preferences derived from (D_p, D_{np}), while ensuring reward retention through supervised fine-tuning and without any environment interaction. This *post-hoc offline safety alignment* ensures π_θ reduces cost constraint violations without sacrificing the reward capabilities of π_{ref}.

Dynamic sampling strategy: We propose a novel way to apply DPO, a preference optimization loss function from i.i.d. language

modeling setting, to the dynamic, sequential decision-making context of Reinforcement Learning. DPO is used for LLM fine-tuning with a preference dataset consisting of triples of the format <*prompt, preferred response, non-preferred response*>. To map DPO to the sequential decision-making setting, for a given state s (analogous to a user prompt) and the associated action a^+ (analogous to a preferred response) in the static offline preference dataset, PREFINE generates a counterfactual action a^- for the state-action pair $(s, a^+) \sim \mathcal{D}_p$ to construct per-state action triples $< s, a^+, a^- >$. The resulting data triples are further utilized by a DPO-inspired loss function. We explain this in detail in Section 3.2. **Data and notation:** We assume offline trajectory logs partitioned into: (i) $\mathcal{D}_p$: safe trajectories with low cost and high reward (preferred), and (ii) $\mathcal{D}_{np}$: unsafe trajectories with high cost (non-preferred). We extract state-action pairs from these for training.

3.2 Practical Implementation for Trajectory Datasets

Computing the DPO loss theoretically requires a preference pair per state, that is, $< s, a^+, a^- >$ [15]. For the counterfactual action in a preference pair, a natural choice is to sample directly from the datasets whenever possible. Concretely, when we wish to draw a preferred pair (s, a^+) from D_p, we can locate the nearest state s' in the opposite dataset (D_{np}) and use the paired action of that state as counterfactual action (a^-). However, in practice, exact matches of high-dimensional states across offline trajectory datasets (e.g., DSRL) are often unavailable. Moreover, if dataset nearest-neighbor counterfactuals are often unrealistic/off-distribution, they produce weak or misleading contrasts. This is further exacerbated by our chosen setting of keeping the size of $\mathcal{D}_{np}$ as small as possible to maintain similarity with real-world scenarios. As a workaround, we sample counterfactual actions from the current policy π_θ simply because policy-sampled alternatives are closer to the decision boundary and therefore more informative. **Justification for using policy-sampled counterfactuals:** Given a trajectory $\mathcal{T} = \{(s_1, a_1), (s_2, a_2), ..., (s_T, a_T)\}$, we want to generate a contrasting trajectory $\mathcal{T}' = \{(s_1, a'_1), (s_2, a'_2), ..., (s_T, a'_T)\}$, where the set of actions $\{a'_i\} \sim \pi_\theta$, such that $c(\mathcal{T}') > c(\mathcal{T})$ if $\mathcal{T} \sim \mathcal{D}_p$ and $c(\mathcal{T}') < c(\mathcal{T})$ if $\mathcal{T} \sim \mathcal{D}_{np}$. Sampling the set of counterfactual actions from the current policy π_θ ensure that those actions reflect actual deployment behavior and creates a progressively harder set of contrasts as π_θ improves, thereby improving the safety signal. Note that the cost of each individual action a'_i sampled from π_θ may not always be higher (when dataset action $a_i \sim \mathcal{D}_p$) or lower ($a_i \sim \mathcal{D}_{np}$) and this may introduce noise in the training process. To avoid the label noise introduced in this manner, using a supervised fine-tuning (SFT) anchor to the reference policy to preserve reward capability helps and it acts as an implicit regularizer. Our ablation (Fig. 6) shows that removing the SFT anchor leads to catastrophic drops in performance, corroborating its necessity. This sampling strategy creates an adaptive self-critique signal where the policy is continually compared against alternatives it would actually propose. Essentially, it sharpens learning as π_θ improves and the sampled actions become more informative. **Note that it is desired that the expected cost of a counterfactual trajectory $\mathcal{T}$ is higher, when $\mathcal{T} \sim \mathcal{D}_p$, and lower when $\mathcal{T} \sim \mathcal{D}_{np}$, and the cost of each individual sampled action does not matter much.**

Interpretation of training-phase trends: Figure2 illustrates how the mean absolute cost gap between dataset and sampled actions evolves during training for preferred and dispreferred trajectories. We calculate the mean absolute cost gap as the absolute value of the difference between the number of cost violations observed for dataset trajectories vs. the trajectories constructed from policy-sampled counterfactual actions. For HalfCheetahVelocity and Walker2dVelocity environments, a cost constraint violation occurs when the horizontal velocity exceeds a certain threshold value. We count the number of such occurrences during the fine-tuning of the reference policy π_{ref} using PREFINE for the first 200K steps. The observed patterns align closely with the expected behavior of PREFINE. For **dispreferred trajectories**, the gap increases steadily from early to late training, indicating that the policy gradually learns to select safer (low-cost) counterfactual actions in regions that were initially unsafe. Early in training, both the dataset and the sampled actions (drawn from the reference policy π_{ref}) exhibit high costs on an average since the reward-optimized policy is not cost-aware which leads to a small gap. As fine-tuning progresses, PREFINE adapts to minimize the average number of cost constraint violations on such high-cost trajectories, resulting in a progressively larger gap, consistent with improved safety alignment.

Conversely, for **preferred trajectories**, we observe the opposite trend: the cost gap decreases over training. Initially, sampled actions incur slightly higher costs on an average than the low-cost dataset actions due to sampling from the reward-optimized reference policy π_{ref}. Note that the average cost of the dataset trajectories is lower in this case so the gap is larger. As training proceeds, however, the policy becomes cost-aware, leading to reduced gaps by the final training phase. Together, these complementary trends demonstrate that the policy sampling-based counterfactual action generation mechanism successfully differentiates between safe and unsafe state regions thereby improving safety and reducing the average number of cost constraint violations. These conclusions assume that the offline dataset affords reasonable local coverage of the policy's state distribution. The preferred and nonpreferred trajectory sets exhibit roughly 70% Jaccard similarity across our tasks and UMAP visualizations (Figure 13) show substantial overlap in visited states.

3.3 Hybrid DPO+SFT Objective with Policy-sampled Actions

We optimize a hybrid objective that combines a DPO-style preference loss with an SFT anchor. Let π_{ref} be the reference policy. Our implemented loss is as follows:

$$\Delta(s, a^+, a^-) = \log \frac{\pi_\theta(a^+|s)}{\pi_{\text{ref}}(a^+|s)} - \log \frac{\pi_\theta(a^-|s)}{\pi_{\text{ref}}(a^-|s)}. \tag{6}$$

$$L_p(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(s,a^+)\sim\mathcal{D}_p,\, a^-\sim\pi_\theta}\Big[\log \sigma\big(\beta\,\Delta(s, a^+, a^-)\big) + \lambda \log \pi_\theta(a^+|s)\Big], \tag{7}$$

$$L_{np}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(s,a^-)\sim\mathcal{D}_{np},\, a^+\sim\pi_\theta}\Big[\log \sigma\big(\beta\,\Delta(s, a^+, a^-)\big)\Big], \tag{8}$$

$$\mathcal{L}_{\text{PREFINE}}(\pi_\theta; \pi_{\text{ref}}) = L_p(\pi_\theta; \pi_{\text{ref}}) + L_{np}(\pi_\theta; \pi_{\text{ref}}). \tag{9}$$

The first expectation in Eq.7 pushes up the relative logit of known-safe a^+ against policy-sampled counterfactual action a^-, while the SFT term $\lambda \log \pi_\theta(a^+|s)$ anchors reward retention; the second expectation in Eq.8

pushes down known-unsafe a^- against policy-sampled counterfactual action a^+. Here, $\lambda(>0)$ balances safety alignment against reward retention and $\beta(>0)$ controls the strength of KL-divergence in the original DPO loss function [15]. Empirically, β and λ are selected via validation set to achieve the desired safety–performance trade-off.

This operationalizes a regularized preference optimization pattern analogous to RPO (DPO loss + SFT loss combination) [12]. The combined objective, $\mathcal{L}_{\text{PREFINE}} = \mathcal{L}_{\text{DPO}} + \lambda\mathcal{L}_{\text{SFT}}$, is functionally identical to RPO's $\mathcal{L}_{\text{RPO}} = \mathcal{L}_{\text{DPO}} + \eta\beta \cdot \mathcal{L}_{\text{SFT}}$, with λ playing the role of $\eta\beta$. Consequently, PREFINE aligns structurally with and is motivated by RPO. While RPO's theory is developed for language models with bandit feedback, we adopt its structural insight for fine-tuning with offline cost preferences in sequential control and empirically validate its benefits here.

In RPO, the SFT loss is a regularizer that prevents the policy from exploiting gaps in the static data coverage (i.e., over-optimization). In Prefine, it serves a dual, even more critical purpose: (i) It acts as a powerful anchor that stabilizes the entire learning process. By constantly pulling the policy π_θ back towards the known-safe behaviors in the dataset D_p. (ii) It ensures that the actions a^+ or a^- sampled from π_θ remain reasonable. Without the SFT anchor, π_θ could drift into a state where the generated actions might be too noisy or nonsensical, completely corrupting the DPO learning signal. We show the importance of including the SFT term in PREFINE's loss function in Fig. 6.

3.4 Efficient Single-Stage Training Procedure

The fine-tuning procedure outlined in Algorithm 1, which requires only a single backward pass per batch i.e., no nested loops or separate models for estimating reward and cost. For trajectory datasets, we construct per-state counterfactual actions via policy sampling (as discussed in Section 3.2) and optimize Eq. (9). This single-stage update circumvents nested optimization loops, yielding up to $8-12\times$ less wall-clock time than cost-based fine-tuning methods (as shown in Section 5), making PREFINE practical for large-scale offline safety-critical tasks. For the exact implementation details used by PREFINE, see Appendix.

Algorithm 1 PREFINE — policy-sampled counterfactual trajectories (algpseudocode)

Require: Preferred trajectory dataset D_p, non-preferred trajectory dataset D_{np}, reference policy π_{ref} (frozen), learnable policy π_θ, batch size B, trajectory length T, hyperparameters β, λ, η

Pretrain π_{ref} via BC on offline dataset and keep it fixed.
for each training iteration **do**
  Sample minibatch of B preferred trajectories $\{\mathcal{T}_i^p\}_{i=1}^B$ from D_p & B non-preferred trajectories $\{\mathcal{T}_j^{np}\}_{j=1}^B$ from D_{np}.
  $\mathcal{T}_i^p = \{(s_{i,t}, a_{i,t}^+)\}$, $\mathcal{T}_j^{np} = \{(s_{j,t}, a_{j,t}^-)\}$
  Initialize empty sets $\mathcal{B}_p \leftarrow \varnothing$ and $\mathcal{B}_{np} \leftarrow \varnothing$.
  for each preferred trajectory $\mathcal{T}_i^p$ **do**
    for $t \leftarrow 1$ **to** T **do**
      Sample counterfactual action $a_{i,t}^- \sim \pi_\theta(\cdot \mid s_{i,t})$.
      Add triple $(s_{i,t},\, a_{i,t}^+,\, a_{i,t}^-)$ to $\mathcal{B}_p$.
    end for
  end for
  for each non-preferred trajectory $\mathcal{T}_j^{np}$ **do**
    for $t \leftarrow 1$ **to** T **do**
      Sample counterfactual action $a_{j,t}^+ \sim \pi_\theta(\cdot \mid s_{j,t})$.
      Add triple $(s_{j,t},\, a_{j,t}^+,\, a_{j,t}^-)$ to $\mathcal{B}_{np}$.
    end for
  end for
  for all $(s, a^+, a^-) \in \mathcal{B}_p \cup \mathcal{B}_{np}$ **do** ▷ Compute for every triple
    $\Delta(s, a^+, a^-) \leftarrow \log \frac{\pi_\theta(a^+|s)}{\pi_{\text{ref}}(a^+|s)} - \log \frac{\pi_\theta(a^-|s)}{\pi_{\text{ref}}(a^-|s)}$.
  end for
  Compute PREFINE loss as per Eq. 9:
  $\theta \leftarrow \theta - \eta\nabla_\theta L_{\text{PREFINE}}$ ▷ Gradient step (Adam)
end for

4 RELATED WORK

Several recent lines of work address safe offline RL and imitation learning via preferences, distribution matching, or explicit constraint modeling. **SafeDPO** [**?**] adapts Direct Preference Optimization with cost-based regularization, but it is fundamentally different from PREFINE: SafeDPO depends on explicit cost signals and is not designed for the highly imbalanced, offline regimes we target. By contrast, the DICE family (e.g., **SafeDICE** [6], [7, 9]) performs stationary distribution matching with learned cost critics and nested or adversarial optimizations—making them natural and meaningful baselines for offline imitation-style objectives, but computationally heavier than our single-stage fine-tuning. For these reasons we evaluate against SafeDICE as a strong offline baseline.

TraC [**?**] uses the same benchmark but addresses a different problem: it actively rebalances preference datasets via relabeling and querying to obtain near-uniform coverage of preferred/dispreferred behaviors. PREFINE explicitly assumes a more challenging setting of fixed, highly *unbalanced* offline dataset with very few unsafe trajectories and performs preference-based fine-tuning without any active data curation; hence a direct comparison to TraC is not warranted because the methods operate under different assumptions and evaluation goals.

Other recent approaches—e.g., latent safety-constrained policy learning [**?**] and constraint-conditioned actor-critic methods [**?**]—seek to encode safety in latent spaces or via constraint-conditioned optimization. PREFINE differs from these by being architecture-agnostic, requiring only a small number of unsafe demonstrations, and using a hybrid DPO–SFT single-stage fine-tuning objective that both preserves task performance and enforces safety alignment efficiently. Together, these distinctions motivate our baseline choices.

5 EXPERIMENTS

In this section, we present an extensive empirical study conducted for PREFINE across various safety-critical control tasks. We specifically address the following three key questions:

Q1 Effectiveness: Can PREFINE meaningfully reduce cost violations compared to other baselines, while preserving the task performance?

Q2 Efficiency: How quickly does PREFINE converge in terms of wall-clock time relative to the baseline methods?

Q3 Robustness & Sensitivity: How does PREFINE respond to preferred datasets ($|\mathcal{D}_p|$) collected at various cost thresholds τ? How does PREFINE's performance vary with the size of preferred dataset ($|\mathcal{D}_p|$), non-preferred dataset ($|\mathcal{D}_{np}|$), and the choice of important hyperparameters.

5.1 Experimental Setup

For our experiments, we use the well-established DSRL benchmark [11] which provides trajectory datasets with wide-ranging rewards and costs for offline safe RL. DSRL datasets have been collected by training multiple policies with SOTA safe RL algorithms with varying cost thresholds. Each dataset consists of a mix of trajectories ranging from safe to highly unsafe generated by different policies, which makes it multimodal in nature. The offline datasets have been collected for 38 tasks across widely recognized environments. We choose 12 tasks out of those due to compute limitations: 7 from SafetyGym [] and 5 from BulletSafetyGym []. To evaluate performance, we follow the constraint variation evaluation protocol proposed in DSRL [11], which is designed to test the adaptability of different algorithms. Specifically, we run each algorithm on every dataset under three separate

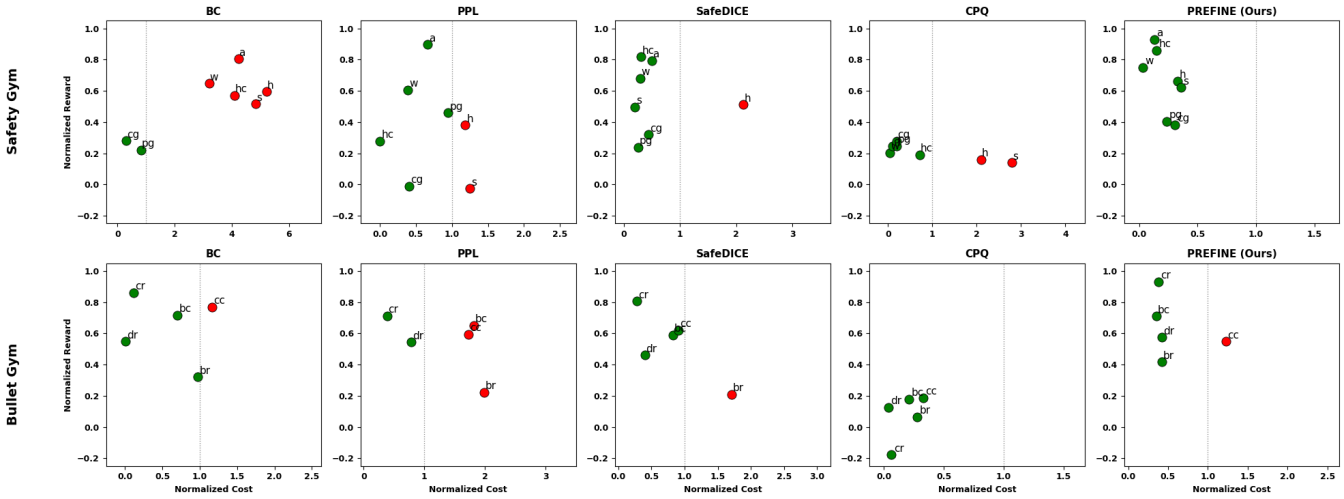

Figure 3: Comparison of PREFINE against baselines in Safety Gym (top) and Bullet Gym (bottom). Each dot denotes a task; green indicates satisfaction of the safety constraint (normalized cost ≤ 1), while red indicates a violation. The vertical dotted line corresponds to the normalized cost threshold of 1. PREFINE consistently concentrates points in the top-left region (high reward, low cost), whereas baselines either violate constraints (BC, PPL, SafeDICE) or trade-off reward for cost (CPQ). Each dot has an abbreviated task label mapped to full task names in Table 1.

Safety Gym

Task	Reference policy (π_{ref})		BC		PPL		SafeDICE		CPQ		PREFINE (Ours)	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓
AntVelocity **(a)**	0.93	8.04	0.81 ± 0.05	4.23 ± 0.06	**0.90 ± 0.04**	**0.66 ± 0.02**	**0.79 ± 0.04**	**0.50 ± 0.02**	**0.24 ± 0.05**	**0.10 ± 0.09**	**0.93 ± 0.04**	**0.13 ± 0.03**
CarGoal **(cg)**	0.44	3.69	**0.28 ± 0.06**	**0.31 ± 0.06**	**-0.01 ± 0.05**	**0.40 ± 0.02**	**0.32 ± 0.05**	**0.44 ± 0.07**	**0.28 ± 0.01**	**0.20 ± 0.04**	**0.38 ± 0.01**	**0.30 ± 0.01**
HalfCheetahVelocity **(hc)**	0.99	8.01	0.57 ± 0.01	4.10 ± 0.04	**0.28 ± 0.01**	**0.00 ± 0.06**	**0.70 ± 0.05**	**0.31 ± 0.04**	**0.19 ± 0.03**	**0.71 ± 0.03**	**0.86 ± 0.05**	**0.15 ± 0.33**
HopperVelocity **(h)**	0.93	8.24	0.60 ± 0.00	5.22 ± 0.04	0.38 ± 0.02	1.19 ± 0.03	0.51 ± 0.08	2.12 ± 0.03	0.16 ± 0.05	2.10 ± 0.08	**0.66 ± 0.03**	**0.33 ± 0.05**
PointGoal **(pg)**	0.64	2.82	**0.22 ± 0.05**	**0.82 ± 0.05**	**0.46 ± 0.01**	**0.95 ± 0.05**	**0.24 ± 0.09**	**0.26 ± 0.00**	**0.25 ± 0.08**	**0.20 ± 0.05**	**0.40 ± 0.03**	**0.24 ± 0.11**
SwimmerVelocity **(s)**	0.94	6.34	0.52 ± 0.07	4.83 ± 0.01	-0.03 ± 0.05	1.25 ± 0.07	**0.50 ± 0.04**	**0.20 ± 0.07**	0.14 ± 0.05	2.79 ± 0.07	**0.62 ± 0.09**	**0.36 ± 0.03**
Walker2dVelocity **(w)**	0.93	9.81	0.65 ± 0.06	3.21 ± 0.04	**0.61 ± 0.05**	**0.39 ± 0.01**	**0.68 ± 0.03**	**0.29 ± 0.04**	**0.20 ± 0.08**	**0.04 ± 0.08**	**0.75 ± 0.01**	**0.03 ± 0.04**

Bullet Gym

Task	Reference policy (π_{ref})		BC		PPL		SafeDICE		CPQ		PREFINE (Ours)	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓
BallCircle **(bc)**	0.89	3.73	**0.72 ± 0.07**	**0.70 ± 0.01**	0.65 ± 0.04	1.82 ± 0.05	**0.59 ± 0.09**	**0.82 ± 0.07**	**0.18 ± 0.04**	**0.21 ± 0.04**	**0.71 ± 0.04**	**0.35 ± 0.10**
BallRun **(br)**	0.95	3.88	**0.32 ± 0.04**	**0.97 ± 0.04**	0.22 ± 0.06	1.99 ± 0.05	0.21 ± 0.04	1.71 ± 0.02	**0.07 ± 0.05**	**0.28 ± 0.04**	**0.41 ± 0.08**	**0.41 ± 0.07**
CarRun **(cr)**	0.99	1.89	**0.86 ± 0.05**	**0.11 ± 0.07**	**0.71 ± 0.04**	**0.39 ± 0.08**	**0.81 ± 0.05**	**0.28 ± 0.07**	**-0.18 ± 0.01**	**0.06 ± 0.06**	**0.93 ± 0.04**	**0.38 ± 0.01**
CarCircle **(cc)**	0.97	4.58	0.77 ± 0.04	1.17 ± 0.02	0.59 ± 0.05	1.73 ± 0.12	**0.62 ± 0.03**	**0.91 ± 0.03**	**0.19 ± 0.06**	**0.33 ± 0.06**	0.54 ± 0.04	1.23 ± 0.04
DroneRun **(dr)**	0.84	6.77	**0.55 ± 0.07**	**0.01 ± 0.07**	**0.54 ± 0.02**	**0.79 ± 0.05**	**0.46 ± 0.05**	**0.40 ± 0.02**	**0.13 ± 0.06**	**0.04 ± 0.02**	**0.57 ± 0.08**	**0.42 ± 0.06**

Table 1: Comprehensive per-task results. Normalized reward (higher is better) and normalized cost (lower is better). Blue indicates the *highest-reward safe agent* per task; bold marks agents with cost less than the cost threshold < 1.

cost thresholds and across five random seeds, allowing for a fair comparison. The evaluation is based on normalized reward and normalized cost [11], where achieving a normalized cost value below 1 signifies that the policy is safe. In the case of implementing PREFINE, we begin by pretraining an expert policy through behavior cloning (BC) using high-reward (irrespective of the cost) trajectories of the offline dataset (as shown in Figure 1), which serves as the reference policy π_{ref} to be used for safety alignment. Next, PREFINE is applied to learn a new policy from the constructed sets of desirable and undesirable trajectories. PREFINE is agnostic to the choice of the underlying training method as long as π_{ref} is differentiable in nature. We use variational autoencoder (VAE) architecture [8] with PREFINE, mainly to capture the multimodality of the DSRL datasets.

Dataset construction: All trajectory subsets (preferred D_p and non-preferred D_{np}) are constructed by a deterministic, pre-specified protocol applied to the same offline task-specific datasets provided by DSRL benchmark suite [11]. For a fixed cost threshold τ we first split the corpus into SAFE $= \{\tau_i \mid \text{cumulative\_cost}(\tau_i) < \tau\}$ and UNSAFE $= \{\tau_i \mid \text{cumulative\_cost}(\tau_i) \geq \tau\}$. To form D_p, we select N_p trajectories from SAFE by stratified sampling across the top reward quantiles (ties resolved by dataset index); to form D_{np} we sample N_{np} trajectories from the bottom 100 trajectories from UNSAFE (sorted by cost) so that D_{np} spans the reward range. The values N_p and N_{np} are fixed prior to model training (we use $N_p = 100$ and $N_{np} = 20$) and the sampling code is published with the released code. We ran experiments with multiple values of N_p and N_{np}, and finally chose the smallest ones that consistently gave good results in multiple tasks. *Note that we present results averaged across three values of τ and 5 different sampled datasets D_p and D_{np} for each value of τ to avoid any bias in dataset construction.* Also note that not all actions within a "safe" trajectory are individually safe; our state-level objective works with expected cost of a trajectory, with SFT regularization mitigating noise from occasional mislabeled steps.

Baselines: We compare against the following baselines: (1) BC: Behaviour Cloning trained on trajectories from preferred dataset $\mathcal{D}_p$ that satisfy safety

constraints. (2) PPL (Bradley-Terry preference learning) [4]: Direct preference learning without SFT anchoring initialized by the reference policy π_{ref} for a fair comparison. (3) SafeDICE [6]: Offline safe imitation learning via distribution matching and cost critics. SafeDICE performs distribution correction explicitly and uses a mixture of unlabelled and labelled trajectory datasets to learn safe behaviour. We modify our datasets in the same way to check SafeDICE performance fairly. (4) CPQ []: a Q-learning based approach that treats unsafe actions as out-of-distribution actions and penalizes them. The hyperparameters for all baselines follow their original publications; code and seeds are fixed across methods to ensure fair comparison. We perform hyperparameter tuning for PREFINE to choose the values of β and λ and use the same values for each group of tasks i.e., SafetyGym and BulletSafetyGym, respectively (see Appendix for more). We intentionally skip comparing PREFINE against other relevant methods such as DWBC [20] and DExperts [13] because SafeDICE already outperforms those [6]. All the experiments are run on an NVIDIA A100 GPU. The results are averaged over 5 different sampled datasets across 3 values of the cost threshold (τ), five random seeds and 100 rollouts at each evaluation step.

5.2 Effectiveness (Q1)

Figure 3 contrasts normalized reward versus normalized cost across tasks. Points to the left of the vertical line (normalized cost = 1) satisfy the safety constraint. PREFINE concentrates in the upper-left region on both suites, indicating simultaneously higher reward and lower cost relative to baselines. In Safety Gym, PREFINE is safe on all tasks while achieving the highest rewards; in Bullet Gym it is safe on most tasks with competitive rewards, exhibiting only a single violation outlier. In contrast, BC and PPL often violate constraints, SafeDICE shows occasional unsafe outliers with reduced reward, and CPQ attains safety (low cost) by substantially under-optimizing reward. These results support our claim that preference-guided fine-tuning from policy samples yields a policy that is safe without being over-conservative.

The main results are presented in Table 1, which reports the performance of all approaches in terms of normalized reward and normalized cost across 12 tasks in two environments averaged over 5 seeds and 3 cost thresholds. PREFINE consistently attains the best trade-off between reward and cost. In Safety Gym, PREFINE achieves the highest average normalized reward (except on PointGoal) while keeping costs well below the safety threshold, clearly outperforming all baselines. In Bullet Gym, PREFINE matches or exceeds the reward levels of the baselines while reducing violations substantially (except on BallRun and CarCircle). When averaged across environments, PREFINE delivers the strongest overall performance. In contrast, other baselines either incur frequent violations (BC, PPL, SafeDICE in Safety Gym) or achieve only marginal rewards while remaining conservative (CPQ).

Regarding safety constraint satisfaction, we present the proportion of tasks solved safely by each approach in Figure 8 (see Appendix). PREFINE consistently achieves the highest fraction of safe tasks across environments, reaching 100% in Safety Gym and 80% in Bullet Gym. In contrast, while baselines such as SafeDICE and CPQ demonstrate moderate levels of safety (83% and 71% in Safety Gym; 60% and 100% in Bullet Gym, respectively), they either fail to generalize across both environments or sacrifice reward excessively to remain safe. Other approaches, including BC and PPL, show much lower safe task fractions and frequently violate cost thresholds. These results highlight that PREFINE is able to combine broad safety coverage with competitive rewards, unlike baselines that trade one for the other.

These results demonstrate PREFINE's ability to retain expert-level behavior while dramatically improving safety. It manages to achieve that without learning a reward model explicitly (like PPL), performing distribution correction and learning a cost model explicitly (like SafeDICE) or performing complex nested optimization to learn a seperate cost model (like CPQ), and without any online environment interactions. This reduction in cost is achieved with minimal loss in task reward. This balance between performance and safety arises from our hybrid DPO+SFT objective: the DPO component sculpts the fine-tuned policy's (π_θ) decision boundary to favor low-cost actions, while the SFT anchor preserves high-reward behaviors learned by the expert policy π_{ref}.

Overall, PREFINE achieves the best balance between reward and safety across all tasks, remaining consistently below the cost threshold. These results highlight PREFINE's practical deployability, as it yields high-performing yet reliably safe policies across diverse environments. We provide the training curves in Appendix.

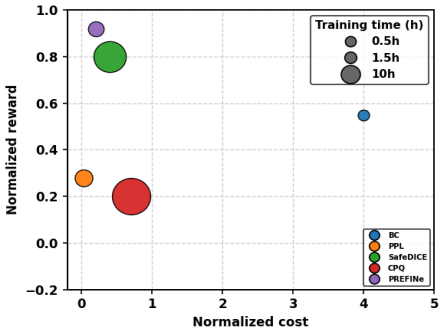

Figure 4: Wall-clock running time (proportional to marker size) comparison of PREFINE with baselines. PREFINE is more scalable.

5.3 Efficiency (Q2)

Efficiency is critical for deploying safety alignment at scale. Figure 4 shows the comparison of PREFINE wall-clock running time with the baselines for the task Walker2dVelocity. The running time is proportional to the marker size. PREFINE achieves a dramatic runtime advantage over competing baselines (e.g., SafeDICE, CPQ). In our experiments PREFINE completes end-to-end fine-tuning in about 1.5 hours, roughly an order of magnitude faster (10× speedup) than CPQ and SafeDICE, which take more than 10 hours to finish running. This improvement stems from PREFINE's single-stage preference fine-tuning (one backward pass per minibatch) versus the nested optimization or distribution-matching updates required by SafeDICE/CPQ. To ensure an apples-to-apples comparison all methods were executed on identical hardware (NVIDIA Tesla V100, 20 GB), used the same dataset splits and random seeds, and shared implementation-level settings (matched network architectures, optimizer, batch size and number of training iterations). We measured wall-clock training time excluding evaluation, used authors' reference code when available or reimplemented baselines using PyTorch with hyperparameters taken from the original papers; these controls ensure the observed timing gap reflects algorithmic cost rather than implementation-based differences.

5.4 Robustness & Sensitivity (Q3)

A key strength of PREFINE is its stability under different cost thresholds and dataset configurations. In Figure 7, we examine the effect of varying the cost threshold τ across multiple tasks. PREFINE maintains consistently high normalized rewards while safety (measured as 1 - cost) remains robust across $\tau = 30, 40, 50$. The cost slightly increases (and safety decreases) as the cost threshold becomes relaxed which is expected. This indicates that PREFINE does not overfit to a particular safety threshold and reliably balances reward and safety across a wide range of cost threshold values.

We also study PREFINE's robustness with respect to dataset size, as shown in Figure 5. On both AntVelocity and Walker2dVelocity, PREFINE achieves stable performance even with relatively small offline datasets. Both normalized rewards and safety stay consistently high. This demonstrates

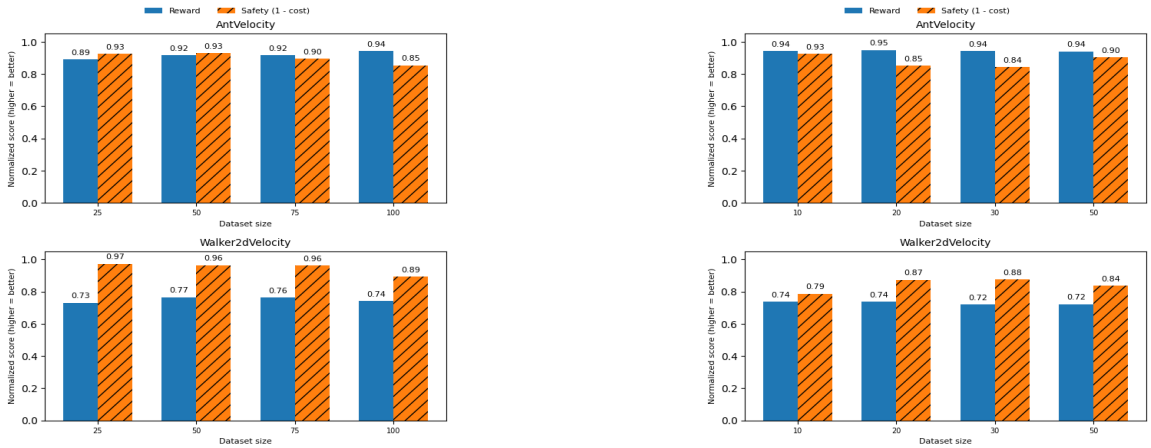

Figure 5: Robustness of PREFINE to dataset size. PREFINE maintains consistently high normalized rewards and strong safety across varying dataset sizes for $\mathcal{D}_p$ (left) and $\mathcal{D}_{np}$ (right), demonstrating stability and data efficiency.

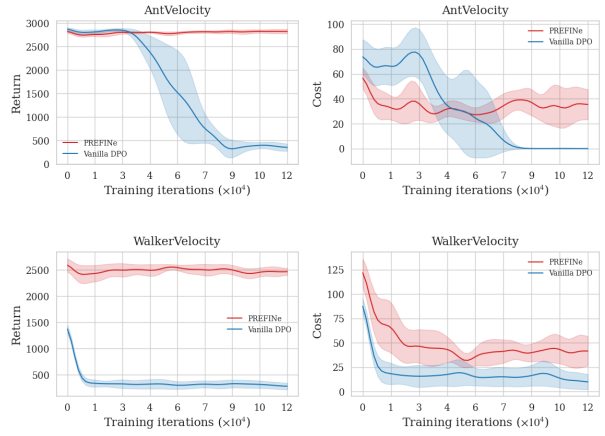

Figure 6: Ablation study: (Left) Safety alignment of reference policy π_{ref} using PREFINE(red) vs vanilla DPO loss(blue). PREFINE uses DPO + SFT loss terms. DPO return (left) rapidly falls down due to unlearning, while PREFINE avoids unsafe behavior while retaining high task performance. Vanilla DPO(blue) shows lower cost in comparison to PREFINE (right) but at the same time, struggles to keep the expert (π_{ref}) performance intact.

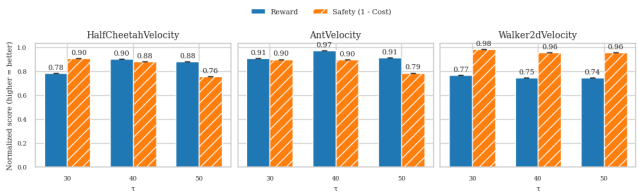

Figure 7: Robustness of PREFINE to cost thresholds. PREFINE maintains consistently high rewards and strong safety across different values of τ, demonstrating stability.

that PREFINE is data-efficient, able to learn safe and rewarding policies even when training data is limited. This quality is especially useful for real-world scenarios where collecting a large number of unsafe demonstrations for training task agents can be challenging.

Next, we sweep the SFT weight λ across $\{0.1, 1.0, 1.6, 2.0\}$ and DPO hyperparameter β across $\{0.05, 0.2, 0.6, 0.95\}$ (see Figure 9 in Appendix). We find that PREFINE performs well for $\lambda \in 1.0, 1.6$ and $\beta \in 0.05, 0.2$ on the tasks we tested. In summary, higher value of λ and a lower value of β work well. Note that β controls the influence of the KL-divergence term in vanilla DPO loss thereby deciding the closeness of the fine-tuned policy to the reference policy. For PREFINE, using a large $beta$ with vanilla DPO loss alone was not working well, as evident by the rapid unlearning effect at the beginning of the fine-tuning process (see Section 5.5).

Discussion These results illustrate a fundamental shift: rather than laboriously engineering cost functions or performing expensive online interactions, we can leverage naturally occurring preferences to retrofit safety into a differentiable pre-trained policy quickly and robustly. PREFINE thus paves a practical path towards safe deployment in domains where data is not easily available and safety is non-negotiable such as autonomous driving systems (reducing collisions) to healthcare (avoiding dangerous drug interactions). However, policy-sampled counterfactual actions for computing DPO loss can introduce noise when sampled actions are not strictly safer/more unsafe. SFT regularization mitigates but does not eliminate this. Future work should explore ways to mitigate adversarial or erroneous feedback.

5.5 Ablation Study

To isolate the role of each component, we conduct an ablation study evaluating the impact of the supervised fine-tuning (SFT) loss $\mathcal{L}_{\text{SFT}}$. We compare PREFINE which combines DPO with SFT against vanilla DPO (Figure 6). **Without the SFT term, the policy reduces constraint violations but suffers a sharp drop in task performance, becoming overly conservative. Return collapses below 88% within the first 10K steps (see Fig. 6).** In contrast, including SFT enables the policy to balance safety and reward, maintaining high returns while enforcing constraints. This highlights SFT as essential for preventing catastrophic unlearning and achieving a robust safety–performance trade-off. Additional ablation results are provided in the Appendix.

6 CONCLUSION

We present PREFINE, a fully offline framework for post-hoc safety alignment of pre-trained differentiable policies. By creating counterfactual trajectories from offline data in continuous domains, PREFINE casts safety as a preference learning problem and makes it amenable to Direct Preference Optimization (DPO) and supervised fine-tuning (SFT). PREFINE eliminates the

need for nested optimization loops and explicit cost estimation. Our empirical evaluation across diverse continuous control and navigation benchmarks demonstrates that PREFINE substantially reduces constraint violations by over 60% while maintaining expert-level task performance. It runs an order of magnitude faster than the baseline methods, completing end-to-end training in under 1.5 hours. Also, it remains robust under limited non-preferred data and wide ranges of hyperparameters. These results underscore PREFINE's potential as a scalable recipe for retrofitting safety into existing policies, paving the way for fast safety alignment.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *ICML*, pages 22–31, 2017.

[2] Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

[3] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[4] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[5] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 4299–4311, 2017.

[6] Youngsoo Jang, Geon-Hyeong Kim, Jongmin Lee, Sungryull Sohn, Byoungjip Kim, Honglak Lee, and Moontae Lee. Safedice: Offline safe imitation learning with non-preferred demonstrations. In *NeurIPS*, volume 36, 2023.

[7] Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *ICLR*, 2022.

[8] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[9] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *ICLR*, 2020.

[10] Yang Liu, Jialin Ding, and Xueqian Liu. Constrained variational policy optimization for safe reinforcement learning. In *ICML*, pages 13644–13658, 2022.

[11] Yang Liu, Jialin Ding, and Xueqian Liu. Dsrl: Benchmarking safe offline reinforcement learning with diverse safety requirements. *arXiv preprint arXiv:2401.14758*, 2024.

[12] Zhihan Liu, Miao Lu, Shenao Zhang, Boyi Liu, Hongyi Guo, Yingxiang Yang, Jose Blanchet, and Zhaoran Wang. Provably mitigating overoptimization in rlhf: Your sft loss is implicitly an adversarial regularizer. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 138663–138697. Curran Associates, Inc., 2024.

[13] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL*, pages 6691–6713, 2022.

[14] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[15] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[16] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *PMLR*, pages 627–635, 2011.

[17] Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 9, pages 1040–1046, 1996.

[18] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *NeurIPS*, pages 11244–11255, 2020.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[20] Yue Wu, Shuangrui Zhai, and Nitish Srivastava. Dwbc: Mitigating catastrophic forgetting in dynamic imitation learning via weight-based consolidation. In *NeurIPS*, volume 35, pages 3722–3734, 2022.

[21] Haoran Xu, Xingyu Zhan, Honglei Yin, and Huiling Qin. Constraints penalized q-learning for safe offline reinforcement learning. In *AAAI*, volume 36, pages 8753–8760, 2022.

A APPENDIX

A.1 DSRL Task Description

We evaluate our approach on the DSRL benchmark [11], a widely adopted suite for studying offline safe reinforcement learning. DSRL offers a comprehensive set of 38 datasets spanning multiple safety-critical environments and difficulty levels. We choose 12 tasks out of SafetyGymnasium [**? ?**] and BulletSafetyGym [**?**] because of compute limitations. Each environment in DSRL is specifically designed to assess the trade-off between task performance and safety in offline learning. We adopt this benchmark because it provides diverse safety constraint types including collision hazards, velocity limits, and dynamic obstacle avoidance allowing for a systematic and fair comparison across methods under heterogeneous safety formulations.

Safety Gym Built on the Mujoco physics engine, SafetyGymnasium provides a variety of safety-aware control tasks. It includes two main agent types, *Car* and *Point*, each associated with four task categories: *Button*, *Circle*, *Goal*, and *Push*. These tasks are further divided into difficulty levels (*1* and *2*) and follow the naming pattern `AgentTaskDifficulty`. Agents must navigate toward goals while avoiding hazards. The benchmark also contains five velocity-constrained Mujoco agents: *Ant*, *HalfCheetah*, *Hopper*, *Walker2d*, and *Swimmer* for studying constraint-driven locomotion. We use all 5 velocity-constrained Mujoco tasks because they run faster along with PointGoal1 and CarGoal1 tasks to maintain constraint diversity.

Bullet Gym Implemented in the PyBullet simulator, BulletSafetyGym follows similar safety principles but features shorter horizons and a broader range of agents. It comprises four agent types: *Ball*, *Car*, *Drone* each performing two task types: *Circle* and *Run*, labeled as `AgentTask`. We choose BallRun, BallCircle, CarRun, CarCircle and DroneRun out of these to cover all agent types.

A.2 Dataset Characteristics

We conduct experiments on Safety Gym and Bullet Gym tasks from DSRL suite with safety constraints. Table 2 shows the Safety Gym tasks, safety constraint settings and the information of dataset for each domain that we used in our experiments. Likewise, Table 3 shows the dataset characteristics for Bullet Gym tasks.

A.3 Evaluation Metrics

To evaluate performance, we follow the protocol established in the DSRL benchmark [11], which reports both *normalized reward* and *normalized cost*. The normalized reward is computed based on the cumulative returns achieved by the policy, while the normalized cost is defined as:

$$C_{\text{normalized}} = \frac{C_\pi + \varepsilon}{\kappa + \varepsilon}, \tag{10}$$

where C_π denotes the cumulative cost under policy π, κ is the predefined cost threshold, and ε is a small positive constant to ensure numerical stability when $\kappa = 0$. Following the DSRL convention, a task is regarded as *safe* when $C_{\text{normalized}} \leq 1$.

A.4 Training Details and Hyperparameters

We adopt a two-step training procedure for PREFINE. First, we pretrain the reference policy π_{ref} using behavior cloning (BC) on high reward trajectories. Next, we refine the policy by applying PREFINE on the newly created preferred and non-preferred datasets. The hyperparameters used in the experiments are summarized below:

Hyperparameters:

- γ (discount factor): 0.99
- Optimizer: Adam
- Learning rate: $3e-4$
- Network size: [256,256]
- Batch size: 256
- Training iterations: 500,000
- Hardware: NVIDIA Tesla V100 GPU
- Fine-tuning duration: ~1.5 hours per task

Other Hyperparameters:

- SFT weight $\lambda \in \{0.1, 1.0, 1.6, 2.0\}$
- DPO temperature $\beta \in \{0.05, 0.2, 0.6, 0.95\}$
- Final values used: $\lambda = 1.6$, $\beta = 0.05$ for Safety Gym; $\lambda = 1.0$, $\beta = 0.2$ for Bullet Gym.

Baseline Hyperparameters: We choose the hyperparameters for PPL, SafeDICE and CQL from their respective papers.

B ADDITIONAL RESULTS

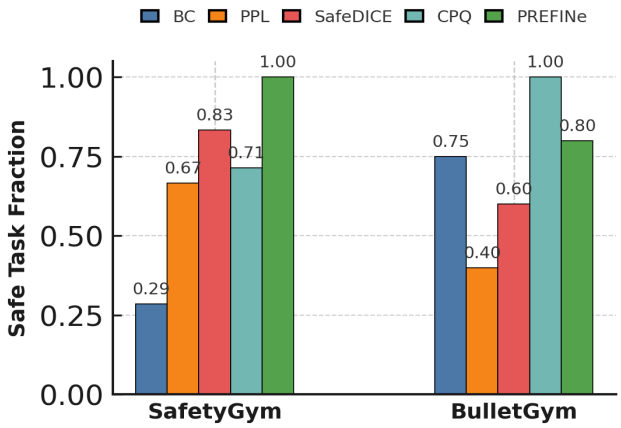

Figure 8: Fraction of tasks solved for safety.

B.1 Fraction of Tasks Solved for Safety

We see in Figure 8 that PREFINE solves the maximum percentage of tasks safely in both Safety Gym and Bullet Gym task suites. In Bullet Gym, it is second only to CPQ and that is because CPQ trades off rewards for safety.

B.2 Ablation Study: PREFINE vs. Vanilla DPO

Vanilla DPO only ensures relative ranking, so it can deviate heavily from expert behavior if it finds safer modes. This leads to catastrophic unlearning—the policy becomes overly conservative or unstable. It is evident by the returns crashing for Vanilla DPO. The SFT term anchors the policy to expert trajectories, acting like a soft KL term and ensuring the task performance doesn't get sacrificed while improving safety. We show the results for Hopper in Figure 10

B.3 Effect of λ and β on Safety Performance Trade-off in Walker

The results in Figure 9 demonstrate a clear trade-off between safety and performance governed by the choice of λ and β. Lower values of λ yield significantly lower cost and CVaR, especially for larger β, but at the expense of reduced reward. Conversely, higher λ improves reward but results in higher safety violations. This highlights the importance of appropriately tuning (λ, β) to balance safety and performance objectives.

B.4 Learning Curves

We provide the learning curves for PREFINE (averaged across three cost threshold values: 30, 40 and 50) in Figure . The results are based on absolute

Table 2: Preference dataset specifications for all domains used in our experimental results on DSRL Mujoco and Safety Gym tasks.

Task specification	Ant	HalfCheetah	Hopper	Walker2d	Swimmer	PointGoal1	CarGoal1
# of preferred demonstrations ($\|\mathcal{D}_p\|$)	100	100	100	100	100	100	100
# of non-preferred demonstrations ($\|\mathcal{D}_{np}\|$)	20	20	20	20	20	20	20
Mean cost of preferred demonstrations	26.15	23.12	26.12	10.15	24.62	9.12	9.04
Mean cost of non-preferred demonstrations	225.10	221.65	225.35	273.35	174.55	88.60	108.25
Mean return of preferred demonstrations	2485.10	2415.50	1616.55	2661.96	132.4	20.30	27.03
Mean return of non-preferred demonstrations	2667.70	2418.10	1157.23	2648.11	102.6	16.47	20.42

Table 3: Preference Dataset specification of each domain used in our experimental results on DSRL Bullet Gym tasks.

Task specification	BallRun	CarRun	BallCircle	CarCircle	DroneRun
Safety Constraint (κ)	80	40	80	100	140
# of preferred demonstrations ($\|\mathcal{D}_p\|$)	100	100	100	100	100
# of non-preferred demonstrations ($\|\mathcal{D}_{np}\|$)	20	20	20	20	20
Mean cost of preferred demonstrations	6.6	34.84	12.54	15.77	11.6
Mean cost of non-preferred demonstrations	74.8	34.84	79.6	92.4	134.22
Mean return of preferred demonstrations	275.8	530.69	740.43	361.47	387.23
Mean return of non-preferred demonstrations	1021.72	505.06	458.36	259.75	481.42

rewards and absolute costs. See Figure 11 for Safety Gym tasks and Figure 12 for Bullet Gym tasks.

B.5 Dataset Overlap

InFigure 13, we show the overlap between preferred and dispreferred dataset states. We find that for all the tasks, the Jaccard similarity between the UMAP embeddings of preferred and non-preferred states comes out to be around 70%. This gives us sufficient confidence to perform counterfactual action sampling from the current policy for PREFINE.

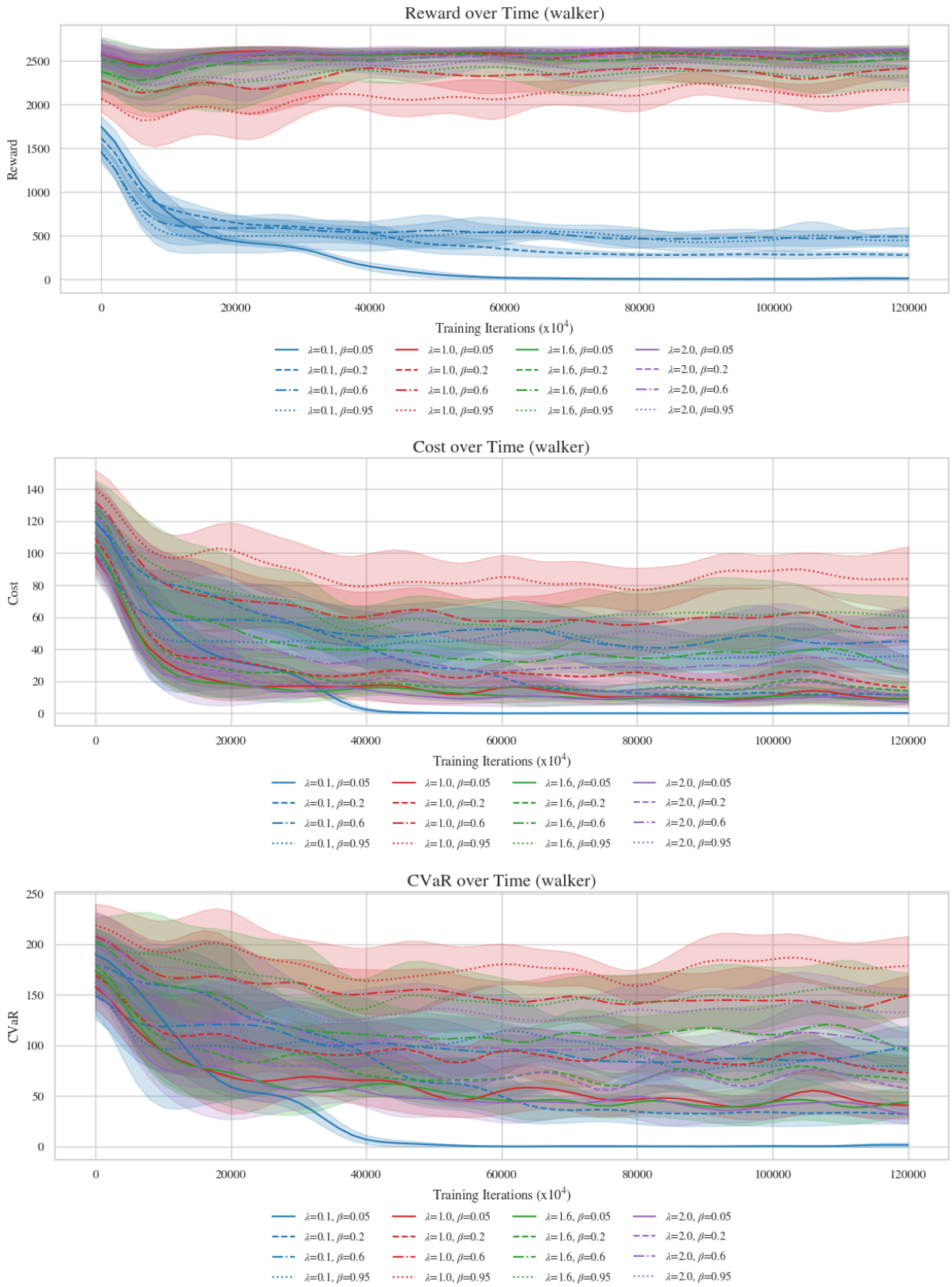

Figure 9: Training dynamics for different values of λ (color) and β (line style) in the Walker environment. Each subplot shows the evolution of a key metric over training iterations: (Top) cost, (Middle) CVaR, and (Bottom) reward. Lower λ emphasizes safety, resulting in reduced cost and CVaR but lower reward, especially at higher β (e.g., $\lambda = 0.1$, $\beta = 0.95$). Higher λ prioritizes reward, often at the expense of safety. Shaded regions denote standard error across three seeds. These plots illustrate the trade-off between safety and performance governed by the (λ, β) configuration.

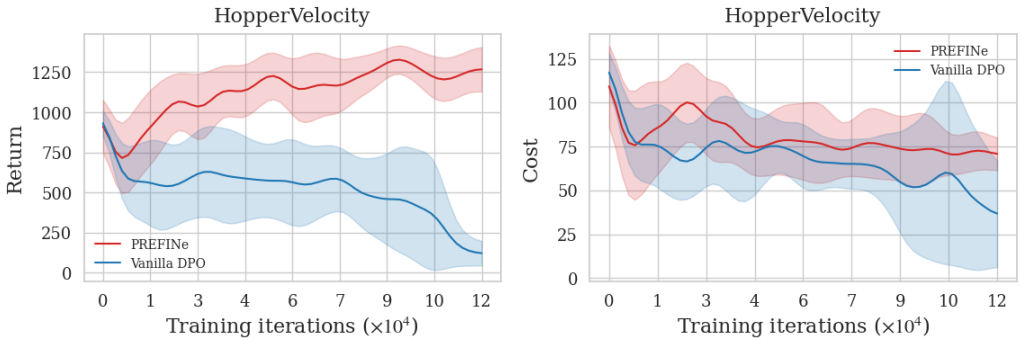

Figure 10: Ablation study 1: (Left) Safety alignment of reference policy π_{ref} using PREFINE(red) vs DPO loss(blue). PREFINE uses DPO + SFT loss terms. DPO return rapidly falls down due to unlearning, while PREFINE avoids unsafe behavior while retaining high task performance. (Right) DPO(blue) shows lower cost in comparison to PREFINE but at the same time, struggles to keep the expert (π_{ref}) performance intact.

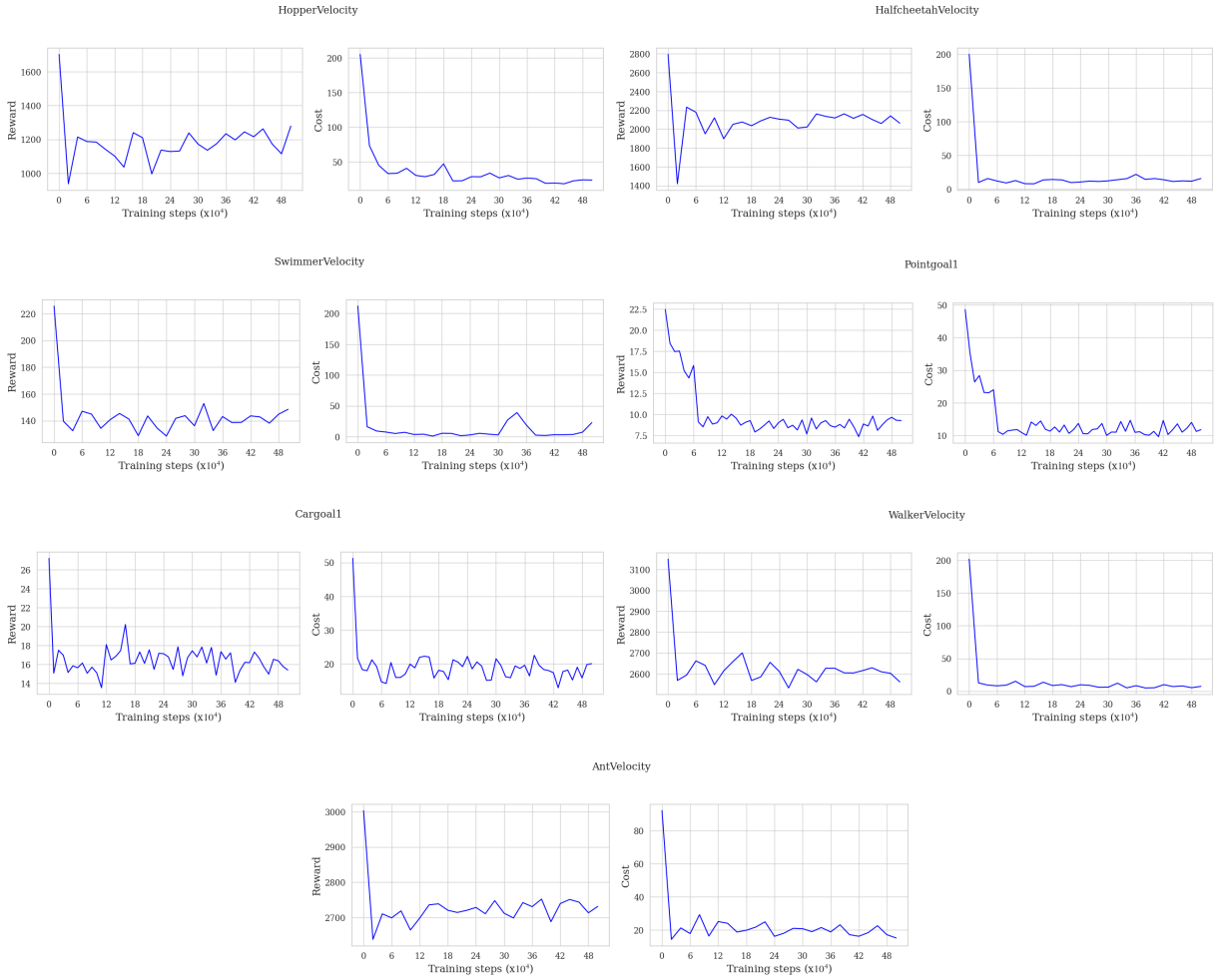

Figure 11: PREFINE training curves for Safety Gym tasks.

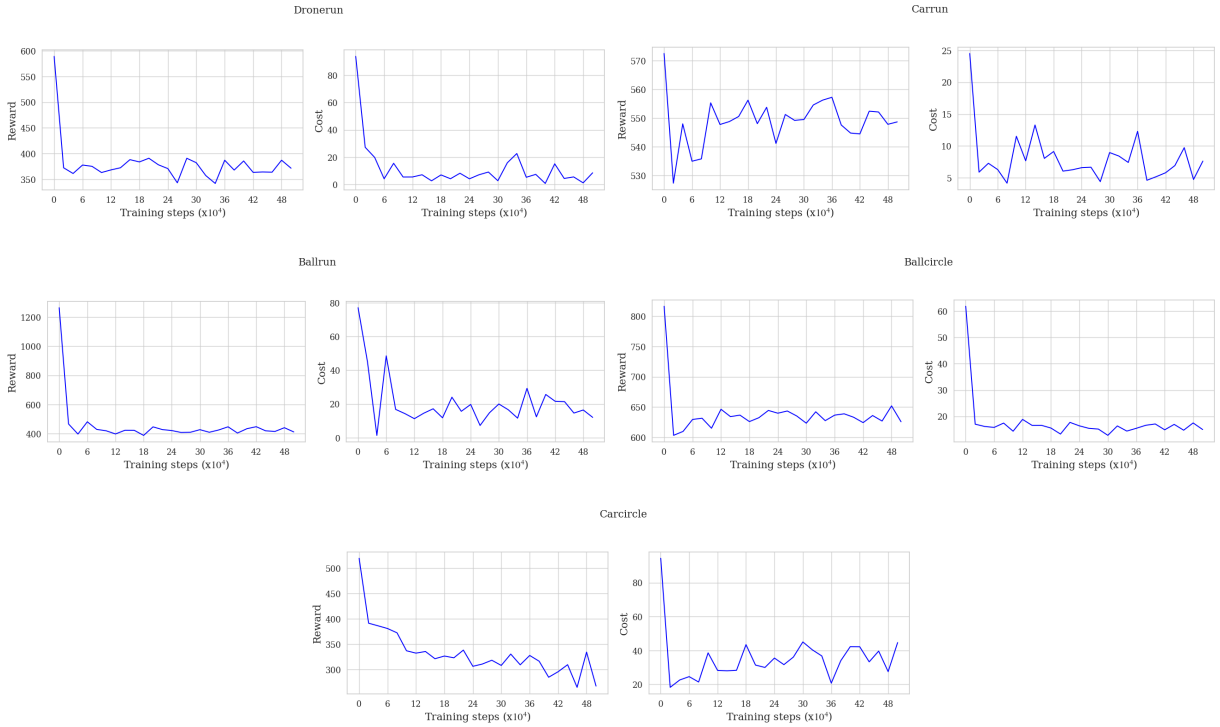

Figure 12: PREFINE training curves for Bullet Gym tasks.

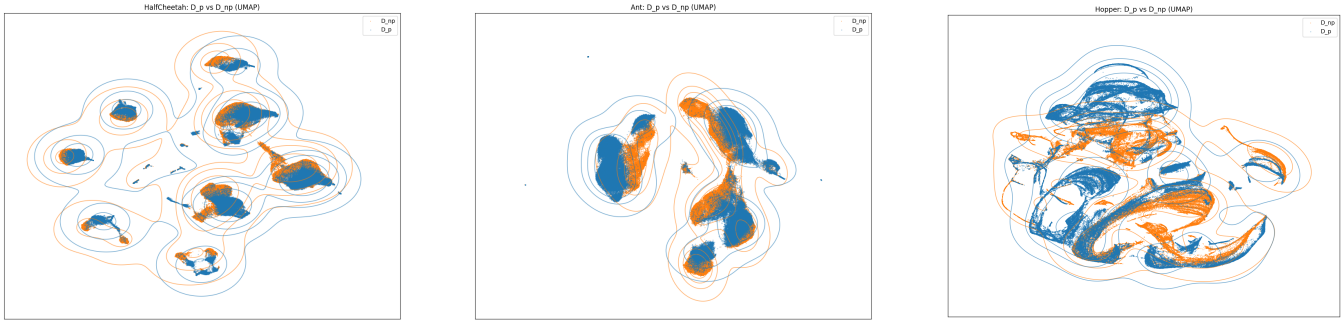

Figure 13: UMAP embeddings of the training datasets used for various tasks showing significant overlap between Preferred dataset states (blue) and non-preferred dataset states (orange).