FLOW-GUIDED LATENT REFINER POLICIES FOR SAFE OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Safe offline reinforcement learning remains challenging due to two coupled obstacles: (i) reconciling soft penalty designs with hard safety requirements, and (ii) avoiding out-of-distribution (OOD) actions when the learned policy departs from the behavior data. Existing approaches often rely on penalty tuning that under- or over-regularizes safety, solve constrained objectives that depend on accurate simulators or online rollouts, or train powerful generative policies that still explore low-density, safety-unknown regions at deployment. We introduce a constraint-free offline framework that addresses both issues by (a) modeling the latent action manifold via a trainable flow-based density conditioned on state to explicitly concentrate probability mass on high-density—and empirically safe—regions, and (b) applying a lightweight refiner stage that performs small, ordered updates in the latent space to jointly improve reward and safety before decoding actions. This design keeps policy search inside the modelled data manifold, while a feasibility-aware training signal steers the refiner toward low-violation solutions without requiring explicit constraints or online interaction. Across various safe offline benchmarks, the proposed method achieves lower violation rates while matching or outperforming baselines in return, demonstrating its potential as a practical and effective approach to safer offline policy learning.

1 INTRODUCTION

Safe offline reinforcement learning (Safe Offline RL) seeks to learn policies that maximize return while satisfying stringent safety requirements from a fixed dataset—without risky, expensive online interaction (Levine et al., 2020). Training from logs allows practitioners to leverage prior operations, simulators, or demonstrations to deploy policies in safety-critical domains (robotics (Wu et al., 2024), driving (Zhang et al., 2025), industrial control (Yu et al., 2025; Wang et al., 2025)) where exploration is untenable, offering clear practical benefits over online learning.

However, simultaneously achieving high performance and strict safety from static data remains elusive (Kushwaha et al., 2025). Most prior work (Ding & Lavaei, 2023; Le et al., 2019; Lee et al., 2022; Fujimoto et al., 2019) encodes safety as soft constraints—risk penalties or constrained Markov decision processes (CMDPs) (Altman, 2021) with Lagrangian updates—so violations are discouraged in expectation. When constraints are tight or objectives conflict, these updates often under-enforce safety, yielding policies that either ignore constraints or require brittle penalty tuning—undesirable in engineering settings that demand near-zero violations. Hard-constraint formulations (Wang et al., 2023; Yu et al., 2022) strengthen safety but typically induce conservatism and measurable return sacrifice, especially offline, where feasible exploration is absent. Orthogonal to constraint design, a second challenge is out-of-distribution (OOD) shift. Offline policies must evaluate and improve without querying unseen actions; otherwise, bootstrapping on OOD actions produces extrapolation error and overestimation, which in turn elevates safety risk at deployment (Kumar et al., 2019). Recent studies further note that OOD states at test time can also degrade behavior, indicating that distribution shift is a coupled state–action phenomenon Levine et al. (2020); Xu et al. (2025).

Optimizing safety and return jointly under OOD constraints is therefore difficult: soft constraints under-enforce safety, hard constraints induce pessimism and missed returns, and generative policies may allocate mass to low-density, safety-unknown regions at inference (Vuong et al., 2022). There-

fore, preventing OOD while improving both objectives remains the central open problem (Kushwaha et al., 2025).

To tackle these challenges, we introduce a density-first, constraint-free framework for safe offline RL, which delivers stringent safety while simultaneously optimizing for high returns. Our approach views safety assurance and OOD control as a representation problem in a task-conditioned latent manifold. Specifically, we equip the critics with Hamilton–Jacobi(HJ)–inspired (Bansal et al., 2017) safety signals: feasibility values are learned with a reversed expectile objective and action-values are updated by an HJ-style backup built from sparse labels, yielding a reliable estimate of feasibility directly from the offline data. On top of this estimator, a pair of structured conditional flow models—one serving as a latent prior and the other as a posterior—sculpts the density so that the projection of actions under a given state concentrates around data-supported regions that are empirically safe. Actions are produced by a high-capacity decoder that is kept fixed during refinement; combined with the coupling structure of the flows, this confines search to a safety-shaped shell in the latent space and suppresses out-of-distribution drift. Building on this manifold, we develop a three-expert (reward, safety, and shared) refiner that performs small, ordered updates in the base latent space with decoupled, advantage-weighted objectives—pushing reward inside feasible regions while pulling away from safety risks in infeasible ones—so that constraint satisfaction and return maximization can proceed jointly in purely offline training.

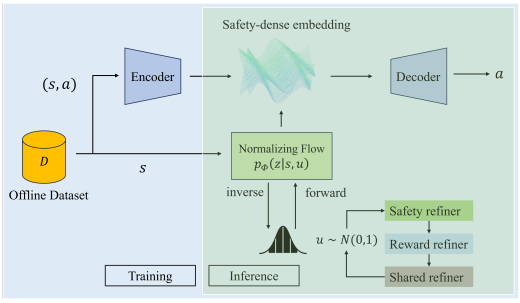


Figure 1: Overview of the proposed method. An encoder maps (s, a) into a safety-dense latent embedding. A conditional normalizing flow $p_\phi(z|, u)$ with base $u \sim \mathcal{N}(0, 1)$ serves as the prior, providing exact forward/inverse transforms between the base and latent spaces; a decoder then reconstructs actions a from z. At inference, three refiners (safety, reward, and a shared refiner) operate in the *base* Gaussian space to adjust samples toward high-density, in-support regions—maximizing return while suppressing OOD actions and enforcing safety constraints.

Furthermore, we provide theoretical guarantees that our refinement procedure constrains distributional shift, offering principled control over OOD actions in the offline setting. Extensive experiments across diverse safe offline benchmarks demonstrate that combining safety-shaped density with latent-space refinement leads to a better return–safety trade-off under hard-constraint scenarios, compared to strong baselines.

2 PRELIMINARIES

Safe offline RL Safe RL is typically formulated as a Constrained Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, h, c, \gamma)$, where $h : \mathcal{S} \to \mathbb{R}$ encodes a state constraint and $c(s) = \max\{h(s), 0\}$ is the induced cost, with $c(s) > 0$ indicating a constraint violation. The discount factor is $\gamma \in (0, 1)$. In the offline setting, we are given a fixed dataset $\mathcal{D} = \{(s, a, r, c, s', d)\}$ collected by an unknown behavior policy π_β. We adopt the basic offline safe RL objective:

$$\max_{\pi} \mathbb{E}_s[V_r^\pi(s)] \quad \text{s.t.} \quad \mathbb{E}_s[V_c^\pi(s)] \le \ell, \qquad D_{\mathrm{KL}}(\pi \,\|\, \pi_\beta) \le \varepsilon, \tag{1}$$

where $V_r^\pi(s) = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\big]$ is the reward value function and $V_c^\pi(s) = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty} \gamma^t c(s_t) \mid s_0 = s\big]$ is the cost value function. ℓ is the cost limit. The KL divergence $D_{\mathrm{KL}}(\cdot\|\cdot)$ constrains the deviation of π from the behavior policy π_β.

Normalizing flows. Normalizing flows (NFs) (Kobyzev et al., 2020) are powerful generative models for complex distribution modeling. Let $u \sim \mathcal{N}(0, I)$ and $z = f_\phi(u; \mathrm{cond})$ be a bijection conditioned on cond (e.g., state or task context). The log-density of z is computed by the change-of-variables formula:

$$\log p_\phi(z \mid \mathrm{cond}) = \log p(u) + \log\Big|\det \frac{\partial u}{\partial z}\Big|. \tag{2}$$

In our implementation, we adopt a RealNVP-style (Dinh et al., 2016) flow architecture based on coupling layers. Each layer splits the input z into two parts: an identity component z_{id} that remains unchanged, and a transform component z_{tr} that is updated through an affine transformation:

$$z'_{\text{tr}} \;=\; z_{\text{tr}} \odot \exp s_\phi\big(z_{\text{id}}, \text{cond}\big) \;+\; t_\phi\big(z_{\text{id}}, \text{cond}\big), \qquad \log|\det J| \;=\; \sum s_\phi(\cdot), \tag{3}$$

where s_ϕ and t_ϕ are scale and translation networks. These layers yield tractable log-likelihoods and exact inverses by construction. Stacking multiple such layers increases the expressiveness of the model while preserving efficient computation due to the triangular structure of the Jacobian.

3 METHODOLOGY

To address the twin challenges of under-enforced soft constraints and out-of-distribution drift in offline settings, we first recast the objective as a *state-wise zero-violation* hard constraint. Concretely, we require $h(s_t) \le 0$ almost surely under $a_t \sim \pi(\cdot \mid s_t)$ for all $t \in \mathbb{N}$. Consequently, the soft safety constraint in Eq. 1 is replaced by a state-wise requirement together with an offline trust region:

$$\max_{\pi} \; \mathbb{E}_s[V_r^\pi(s)] \quad \text{s.t.} \quad V_c^\pi(s) \;\le\; 0, \qquad \mathrm{D}_{\mathrm{KL}}(\pi \,\|\, \pi_\beta) \;\le\; \varepsilon. \tag{4}$$

Building on this formulation, we adopt a base-space refinement strategy, where optimization is performed entirely within a conditional latent density that is confined to a data-supported manifold, as illustrated in Figure 1. We refer to our method as **FLRP**—*Flow-guided Latent Refiner Policies*—which enables in-distribution policy improvement with near-zero constraint violations. The core components of FLRP are detailed in the following subsections.

3.1 FEASIBILITY-BASED VALUE FUNCTION

The state-wise zero-violation requirement in Eq. 4 calls for a representation that certifies safety along the entire trajectory, not only in expectation. Hamilton–Jacobi (HJ) reachability Bansal et al. (2017) from safe control provides exactly such a representation through signed safety functions and value-based certificates, and has been shown to be effective for enforcing hard constraints in recent safe RL studies (Fisac et al., 2018; Yu et al., 2022). Following this line, we cast the hard constraint into a pair of feasibility value functions based on Definition 1 that we can learn from offline data and then use as a unified signal for policy generation and refinement.

Definition 1 (Optimal feasible value functions)**.** *Let $h : \mathcal{S} \to \mathbb{R}$ be a signed safety function with $h(s) \le 0$ denoting safety. The optimal state-wise and action-wise feasibility values are defined by*

$$V_h^\star(s) := \min_{\pi} \max_{t\in\mathbb{N}} h(s_t), \quad s_0 = s,\; a_t \sim \pi(\cdot \mid s_t),\; s_{t+1} \sim T(\cdot \mid s_t, a_t), \tag{5}$$

$$Q_h^\star(s,a) := \min_{\pi} \max_{t\in\mathbb{N}} h(s_t), \quad s_0 = s,\; a_0 = a,\; a_{t\ge 1} \sim \pi(\cdot \mid s_t). \tag{6}$$

By construction, $V_h^\star(s) \le 0$ implies the existence of a policy whose entire trajectory from s remains safe; likewise, $Q_h^\star(s,a) \le 0$ certifies zero violations when starting with action a. In offline settings, they can be estimated by the Feasible Bellman Operator with a discounted factor γ.

Definition 2 (Feasible Bellman operator)**.** *For $\gamma \in (0,1)$ and any $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the feasible Bellman operator is defined by*

$$(\mathcal{P}^\star Q)(s,a) := (1-\gamma)\, h(s) \;+\; \gamma max\{\, h(s),\; V^\star(s')\,\}, \quad V^\star(s') := \min_{a'} Q(s',a')\,. \tag{7}$$

This operator is a γ-contraction under the sup norm and admits a unique fixed point $Q_{h,\gamma}^\star$ with $V_{h,\gamma}^\star(s) = \min_a Q_{h,\gamma}^\star(s,a)$; as $\gamma \uparrow 1$, it recovers the HJ-style values $Q_h^\star$ and $V_h^\star$ in Definition 1. Proof is deferred to Appendix B.2.

We parameterize (Q_h, V_h) with neural networks. To avoid extrapolation errors that arise from querying actions outside the data support (Fujimoto et al., 2019), we approximate $Q_h(s,\cdot)$ by reversed expectile regression and train Q_h with a one-step target that uses V_h in place of $\min_{a'} Q_h(s',a')$:

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[\rho_{\tau_h}^{\text{rev}}\big(Q_h(s,a) - V_h(s)\big)\big]\,, \tag{8}$$

$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\Big[\Big((1-\gamma)h(s) + \gamma \max\{h(s), V_h^{\text{tgt}}(s')\} - Q_h(s,a)\Big)^2\Big]\,. \tag{9}$$

where $\rho_\tau^{\text{rev}}(u) = \left|\tau - \mathbf{1}\{u > 0\}\right| u^2$ and V_h^{tgt} is a slowly updated target network. The reversed expectile with $\tau_h \in (0.5, 1)$ down-weights overly optimistic Q_h values and sharpens the zero level set $V_h \approx 0$, while the target network stabilizes bootstrapping.

3.2 Conditional Flow-based Safe Policy Generation

Rather than learning a policy directly in action space, we model a conditional latent action distribution, where high-quality samples correspond to higher density. Thus, instead of being pushed by hard constraints, safety is pulled by density. Given the empirical feasibility signals learned in Sec. 3.1, we instantiate a conditional flow prior/posterior with a decoder. Compared with other generative models, normalizing flows offer exact likelihood, tractable inverse mapping, and strong expressivity (Papamakarios et al., 2021)—making them well-suited for both density modeling and OOD control.

Safety-weighted ELBO. Let $u \sim \mathcal{N}(0, I)$ be a base latent vector. The prior flow maps u to a latent variable $z = f_\phi(u; s)$, where the log-density is tractable:

$$\log p_\phi(z \mid s) = \log p(u) + \log\left|\det \frac{\partial u}{\partial z}\right|. \tag{10}$$

The posterior flow $q_\psi(z \mid s, a)$ serves as an amortized recognizer, while a decoder $\pi_\theta(a \mid s, z)$ maps latent codes back to actions. Training follows a safety-weighted variational objective that encourages accurate reconstruction and alignment with the prior:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\mathbb{E}_{z\sim q_\psi}\left[-\, w(s,a)\, \log \pi_\theta(a \mid s, z)\right] + \beta\, \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w(s,a)\, \mathrm{D}_{\text{KL}}\left(q_\psi \,\|\, p_\phi\right)\right], \tag{11}$$

where $w(s,a) = \sigma(-Q_h(s,a)/T_q)\ \sigma(-V_h(s)/T_v)$ is a feasibility-weighted score derived from the critics in Sec. 3.1, T_v and T_q are temeperatures, and σ is the logistic function. We formally justify that the above objective remains a consistent variational estimator by showing that it performs a KL projection of the model joint distribution onto a safety-weighted behavior distribution, as stated in the following lemma.

Lemma 1. *Let $\tilde{p}_\mathcal{D}(s,a) \propto w(s,a)\, p_\mathcal{D}(s,a)$ be a behavior-weighted empirical distribution. Then*

$$\mathcal{L}_{\textit{ELBO}} = \text{const} + D_{\text{KL}}\left(\tilde{p}_\mathcal{D}(s,a)\, q_\psi(z \mid s, a) \,\middle\|\, p_\phi(z \mid s)\, \pi_\theta(a \mid s, z)\right).$$

This result shows that $\mathcal{L}_{\textit{flow}}$ amounts to a KL projection of the behavior-weighted posterior onto the generative model distribution. The proof is provided in Appendix B.2.

Prior Density Shaping. Compared to a Gaussian prior, the flow-based prior is capable of modeling more complex and multimodal latent structures, but this expressiveness also introduces challenges during training. To mitigate these difficulties, we introduce a regularization objective that encourages empirically feasible regions in the action space to be mapped back to high-density regions in the latent base space. A key advantage of normalizing flows is their ability to compute an exact inverse transformation from z to u. We leverage this to define the following prior-shaping loss:

$$\mathcal{L}_{\text{shape}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp(Q_r(s,a) - V_r(s)/\beta_r) \cdot \mathbf{I}_{\text{feas}}(s,a) \cdot \left\|T_\phi^{-1}(z_q \mid s)\right\|^2\right] \tag{12}$$

Here, $\mathbf{I}_{\text{feas}}(s,a) = \mathbf{1}\{Q_h(s,a) \le 0\}$ is a binary indicator derived from the feasibility critic, and $T_\phi^{-1}(z_q \mid s)$ denotes the inverse transformation that maps a decoded action back to the latent base space. This encourages the flow prior to assign higher and smoother base-space density to actions that are both safe and high-reward, thereby shaping the latent manifold to better align with feasible and desirable behaviors.

Freezing the decoder and distribution shift. At inference time, actions are generated by sampling $u \sim \mathcal{N}(0, I)$, transforming it through the prior flow $z = f_\phi(u; s)$, and decoding via $a = \pi_\theta(z, s)$. In the subsequent refinement stage (Sec. 3.3), the decoder π_θ is frozen and only u is updated. This confines policy updates to the safety-shaped latent manifold and avoids reintroducing distribution shift through unconstrained decoding.

We show in the following that, under a fixed decoder, the divergence between the learned policy and the behavior policy can be decomposed into controllable terms.

Lemma 2. *Let $\pi_0(\cdot|s) := T_{s\#}\mathcal{N}$ be the action distribution obtained by pushing the standard Gaussian through the frozen prior and decoder, and $\Pi_\theta(a \mid s)$ denotes the learned policy distribution (after refinement). Assume absolute continuity and a bounded density ratio $R_\theta(s) := \sup_a \frac{\pi_0(a|s)}{\pi_\beta(a|s)} < \infty$ on the data support. Then for any state s (proofs are in Appendix B.3),*

$$D_{\mathrm{KL}}\big(\Pi_\theta(\cdot|s) \,\|\, \pi_\beta(\cdot|s)\big) \;\le\; D_{\mathrm{KL}}\big(\Pi_\theta(\cdot|s) \,\|\, \pi_0(\cdot|s)\big) \;+\; \log R_\theta(s).$$

Moreover, by data-processing inequality (DPI) (Beaudry & Renner, 2011) and flow invariance, $D_{\mathrm{KL}}(\Pi_\theta\|\pi_0) \le D_{\mathrm{KL}}(q_u\|\mathcal{N})$, hence $D_{\mathrm{KL}}(\Pi_\theta\|\pi_\beta) \le D_{\mathrm{KL}}(q_u\|\mathcal{N}) + \log R_\theta(s)$.

This result shows that the decoder decouples policy shifts into (i) a base-space divergence term and (ii) a modeling error term, both of which can be controlled during training.

Full objective. We summarize the flow module's objective as:

$$\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{ELBO}} + \mathcal{L}_{\text{shape}} + \lambda_H\big(H_0 - \mathbb{E}_{q_\psi}[-\log q_\psi(z \mid s, a)]\big)_+, \tag{13}$$

where the final term softly enforces a minimum posterior entropy to prevent mode collapse. Having shaped a structured latent manifold through feasibility-aware density modeling, we next develop a refiner module that further improves performance by optimizing within this base space.

3.3 Base-space Optimization via Expert Refiner

While the flow module already shapes a safety-aware latent manifold, it does not directly optimize task performance, as a high reward is also desired. Inspired by recent progress on Mixture-of-Experts (MoE) (Jayawardana et al., 2025; Obando-Ceron et al., 2024) architectures, we design an expert refiner that operates on the Gaussian base latent $u \sim \mathcal{N}(0, I)$ learned in Sec. 3.2. The refiner performs small, ordered updates in the base space to improve reward while keeping search confined to the safety-shaped manifold.

Architecture. The refiner consists of three latent-space experts: a reward expert f_r, a safety expert f_h, and a shared expert f_{sh}. Each expert applies a residual update in the latent base space conditioned on the state s. At each refinement step $t = 0, \ldots, T-1$, we start from $u_0 \sim \mathcal{N}(0, I)$ and apply the following sequential updates:

$$u_{t+1} = u_t + f_k(s, u_t), \quad \text{for } k \in \{r, h, \text{sh}\},$$

where the final update is always performed by the shared expert f_{sh}. After T steps, the refined latent u_T is mapped to $z = f_\phi(u_T; s)$ via the frozen prior flow, and then decoded to an action distribution $\pi_\theta(\cdot \mid s, z)$ using the decoder. We denote its decoded mean by $\bar{a}(s, u_T)$ and use it for downstream evaluation or rollouts.

Expert-specific objectives. Let $\bar{a}(s, u_T) := \arg\max_a \pi_\theta(a \mid s, f_\phi(u_T; s))$ denote the decoded mean action, and reuse the learned critics (Q_r, V_r) and (Q_h, V_h) from Sec. 3.1. Each expert is trained using a modular, advantage-weighted regression (AWR) (Peng et al., 2019; Hansen-Estruch et al., 2023) objective:

(i) Safety expert. Minimizes the violation gap with a push–pull form:

$$\mathcal{L}_h \;=\; \mathbb{E}_{s\sim\mathcal{D}}[w_h(s) \;\cdot \phi\big(Q_h(s, \bar{a}(s, u_T)) - V_h(s)\big)]. \tag{14}$$

where $w_h(s) \;=\; \exp\big(\,[Q_h(s, \bar{a}) - V_h(s)]/\beta_h\big)$ emphasizes samples on truly unsafe moves (positive advantage), and $\phi(\cdot)$ is a soft penalty (e.g., softplus).

(iI) Reward expert. Maximizes return within feasible states as a supervised learning:

$$\mathcal{L}_r = -\,\mathbb{E}_{s,a\sim\mathcal{D}}\left[w_r(s,a) \;\cdot ||\bar{a}(s, u_T) - a||_2\right]. \tag{15}$$

where $w_r(s,a) \;=\; \exp\big(\,[\,Q_r(s,a) - V_r(s)\,]/\beta_r\big) \cdot \mathbf{I}_{\text{feas}}$ up-weights positive reward advantage and prevents reward-only updates from steering into unsafe states.

(iii) Shared expert. Regularizes refinement in the *base* space. As stated in Lemma 2: once the decoder is frozen, the policy shift is entirely induced by the divergence of the refined base distribution $D_{\mathrm{KL}}(q_u\|\mathcal{N})$. Considering the base is a standard Gaussian distribution, we use its energy as an explicit regularizer, together with a small proximal term that discourages large steps:

$$\mathcal{L}_{\text{sh}} = \|u_T\|^2 + \|u_T - u_0\|^2. \tag{16}$$

The full refiner loss is:

$$\mathcal{L}_{\text{ref}} = \lambda_r \mathcal{L}_r + \lambda_h \mathcal{L}_h + \lambda_{\text{sh}} \mathcal{L}_{\text{sh}} \tag{17}$$

Refining in the base space with a fixed process provides distributional control for *all* downstream spaces. Because the flow and decoder are both invertible or frozen, any change in the base space deterministically propagates through the latent and action spaces. While Lemma 2 establishes a general data-processing inequality under pushforward mappings, we now apply this result specifically to our architecture. The next lemma formalizes the KL chain via pushforwards in our method.

Lemma 3. *Let q_u be the refined base distribution and $\mathcal{N}$ the standard Gaussian. Let $f_\phi(\cdot; s)$ be the (invertible) flow and $q_z = f_{\phi\#} q_u$, $p_\phi = f_{\phi\#}\mathcal{N}$, and action distributions $\pi = T_{s\#} q_u$, $\pi_0 = T_{s\#}\mathcal{N}$ with $T_s(u) := \bar{a}(s, u)$. Then (proofs are in Appendix B.4):*

$$D_{\text{KL}}\big(\pi(\cdot \mid s) \,\|\, \pi_0(\cdot \mid s)\big) \;\le\; D_{\text{KL}}\big(q_z \,\|\, p_\phi\big) \;=\; D_{\text{KL}}\big(q_u \,\|\, \mathcal{N}\big). \tag{18}$$

The equality follows from the invariance of KL under invertible mappings (the flow), and the inequality is the data-processing inequality through the decoder.

Corollary 1 (Deviation bounds from base KL)**.** *Let L_g be the Lipschitz constant of g_θ on the latent chart, $W_2(\cdot, \cdot)$ denotes the 2-Wasserstein distance, and $\text{TV}(\cdot, \cdot)$ stands for total variation distance between distributions. Then for any s (proofs are in Appendix B.5):*

$$\begin{aligned} W_2\big(\pi, \pi_0\big) &\le L_g \sqrt{2\, D_{\text{KL}}(q_u \,\|\, \mathcal{N})} \\ \text{TV}(\pi, \pi_\beta) &\le \sqrt{\tfrac{1}{2}\, D_{\text{KL}}(\pi \,\|\, \pi_0)} + \text{TV}(\pi_0, \pi_\beta) \end{aligned} \tag{19}$$

and for any measurable OOD region $\mathcal{O}$:

$$\pi(\mathcal{O}) \le \pi_\beta(\mathcal{O}) + \sqrt{\tfrac{1}{2}\, D_{\text{KL}}(q_u \,\|\, \mathcal{N})} + \text{TV}(\pi_0, \pi_\beta). \tag{20}$$

These results justify our design: keeping $D_{\text{KL}}(q_u \,\|\, \mathcal{N})$ small suffices to bound downstream deviation—latent, action, and final policy-across multiple metrics, while perturbations in z or a lack such guarantees. In practice, although flow architectures are highly expressive, they often make optimization in the latent space z difficult and unstable. To address this, our loss design encourages the mapping from latent space to base space to concentrate mass in high-density regions. This allows us to perform optimization in the base space, where updates remain tractable while still inducing meaningful changes in the latent space.

3.4 Practical Implementation

We employ expectile regression to obtain in-sample, asymmetric value estimates that are biased toward high-value actions without querying out-of-distribution actions, following the practice in IQL (Kostrikov et al., 2021), which trains V_r using asymmetric expectile regression and Q_r by TD updates toward V_r.

$$\mathcal{L}_{V_r} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[\rho_{\tau_r}\big(Q_r(s,a) - V_r(s)\big)\big], \quad \rho_\tau(u) = \big|\tau - \mathbf{1}\{u < 0\}\big|\, u^2, \tag{21}$$

$$\mathcal{L}_{Q_r} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\Big[\big(Q_r(s,a) - \hat{Q}_r(s,a)\big)^2\Big], \quad \hat{Q}_r(s,a) := r(s,a) + \gamma V_r(s'). \tag{22}$$

The overall training procedure proceeds in phases. First, we train the safety critics (Q_h, V_h) using offline data. Then we train the flow module (prior, posterior, decoder) using the safety-weighted ELBO and density shaping objectives in Sec. 3.2. Finally, we train the expert refiner in base space via AWR-style updates to reward and safety. All components are updated purely offline using transitions from the dataset, with no interaction with the environment. This modular design allows each component to focus on its role—feasibility shaping, density modelling, or reward refinement—while maintaining a consistent, in-distribution optimization flow.

At inference time, we sample $u \sim \mathcal{N}(0, I)$, apply the expert refiner for T steps to obtain u_T, decode through the frozen flow and decoder to obtain the final action. Training details and pseudocode can be found in Appendix C.5.

Table 1: Performance Comparison on DSRL benchmark. ↑ means the higher the better, ↓ means the lower the better.

Task	BCQL		CPQ		CDT		FISOR		LSPC		FLRP(Ours)	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
Safety-Gymnasium												
CarButton1	0.16	4.20	0.13	2.44	0.21	1.60	**-0.04**	**0.58**	**-0.15**	**0.58**	**0.03**	**0.36**
CarButton2	0.07	3.47	0.17	7.05	0.13	1.58	**-0.01**	**0.22**	**-0.03**	**0.59**	**0.04**	**0.38**
CarPush1	**0.09**	**0.56**	**-0.14**	**0.80**	**0.31**	**0.40**	0.26	1.23	**0.21**	**0.13**	**0.20**	**0.04**
CarPush2	**0.06**	**0.61**	0.10	5.66	0.19	1.30	**0.16**	**0.71**	0.04	1.37	**0.24**	**0.36**
CarGoal1	**0.13**	**0.90**	**0.22**	**0.79**	0.66	1.21	**0.42**	**0.88**	**0.23**	**0.71**	**0.27**	**0.00**
CarGoal2	0.13	2.38	0.17	3.10	0.48	1.25	**0.06**	**0.06**	**0.11**	**0.50**	**0.20**	**0.28**
AntVel	0.29	2.08	**-0.31**	**0.00**	**0.98**	**0.39**	**0.90**	**0.00**	**0.91**	**0.02**	**0.69**	**0.00**
HalfCheetahVel	1.04	7.06	0.08	2.56	**0.97**	**0.55**	**0.88**	**0.00**	**0.86**	**0.18**	**0.94**	**0.16**
SwimmerVel	0.29	4.10	0.31	2.66	0.67	1.47	**0.01**	**0.01**	0.47	1.26	**0.06**	**0.00**
Average	0.25	2.82	0.08	2.78	0.51	1.08	**0.29**	**0.40**	**0.29**	**0.59**	**0.33**	**0.18**
Bullet-Safety-Gym												
AntRun	0.05	4.63	**0.13**	**0.01**	0.69	1.24	**0.45**	**0.76**	0.94	1.46	**0.52**	**0.00**
BallRun	**0.35**	**0.20**	0.85	13.67	**0.88**	**0.86**	**0.14**	**0.00**	**0.08**	**0.00**	**0.16**	**0.00**
CarRun	0.75	2.51	**0.75**	**0.52**	0.99	1.47	**0.80**	**0.00**	**0.75**	**0.22**	**0.87**	**0.00**
DroneRun	**0.65**	**0.71**	**0.26**	**0.44**	**0.71**	**0.60**	**0.41**	**0.57**	0.62	1.34	**0.59**	**0.02**
AntCircle	0.61	1.42	**0.00**	**0.00**	0.46	2.74	**0.23**	**0.00**	**0.40**	**0.78**	**0.45**	**0.25**
BallCircle	0.79	1.20	0.40	4.37	0.79	1.64	**0.45**	**0.00**	0.29	1.83	**0.46**	**0.00**
CarCircle	0.64	1.80	0.49	4.48	0.70	1.20	**0.34**	**0.00**	**0.28**	**0.04**	**0.66**	**0.06**
DroneCircle	0.68	1.19	-0.27	1.29	0.59	1.56	**0.60**	**0.00**	0.66	1.37	**0.54**	**0.00**
Average	0.57	1.71	2.50	3.10	0.73	1.41	**0.43**	**0.17**	**0.50**	**0.88**	**0.54**	**0.04**
Safe MetaDrive												
Easysparse	0.94	9.25	**-0.05**	**0.15**	**0.25**	**0.15**	**0.41**	**0.50**	0.74	1.55	**0.32**	**0.20**
Easymean	0.99	7.22	**-0.06**	**0.00**	**0.42**	**0.25**	**0.43**	**0.67**	**0.70**	**0.68**	**0.25**	**0.10**
Easydense	0.20	1.76	**-0.06**	**0.16**	0.35	1.17	0.52	1.26	0.74	1.48	**0.33**	**0.11**
Mediumsparse	0.94	2.83	**-0.08**	**0.12**	0.78	1.24	**0.43**	**0.08**	**0.97**	**0.79**	**0.31**	**0.06**
Mediummean	0.70	4.45	**-0.07**	**0.16**	0.72	2.74	**0.36**	**0.02**	**0.92**	**0.89**	**0.52**	**0.63**
Mediumdense	0.76	3.90	**-0.08**	**0.10**	0.70	2.62	**0.51**	**0.39**	**0.87**	**0.88**	**0.33**	**0.07**
Hardsparse	0.49	3.16	**-0.05**	**0.10**	**0.26**	**0.46**	**0.33**	**0.24**	0.52	1.32	**0.35**	**0.34**
Hardmean	0.29	3.80	**-0.05**	**0.15**	**0.20**	**0.61**	**0.27**	**0.01**	**0.41**	**0.57**	**0.28**	**0.10**
Harddense	0.42	2.95	**-0.04**	**0.12**	0.22	1.38	**0.30**	**0.26**	0.53	1.63	**0.36**	**0.11**
Average	0.64	4.37	**-0.06**	**0.12**	0.45	1.18	**0.40**	**0.38**	0.71	1.09	**0.34**	**0.19**

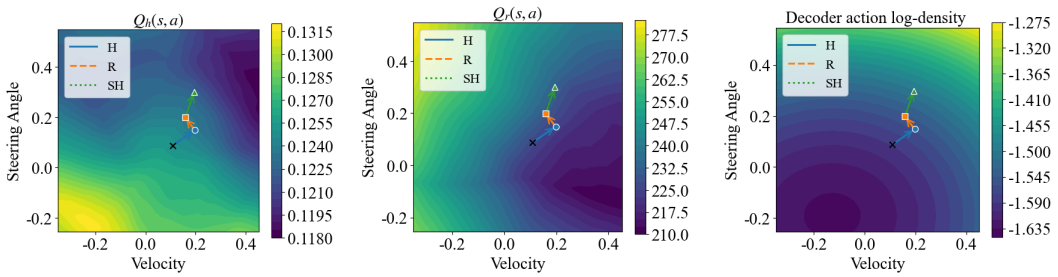
Note: **Bold**: safe policy; Gray: unsafe policy; **Bold blue**: best safe policy; **Bold**: second best safe policy

4 EXPERIMENTS

Experiment Setup. We evaluate the proposed method against several strong offline safe RL baselines across two widely-used benchmark environments: **Safety-Gymnasium** (Ji et al., 2023), **Bullet-Safety-Gym** (Gronauer, 2022) and **Safe Metadrive** (Li et al., 2022) from the DSRL suite (Liu et al., 2023a). We adopt *normalized return* and *normalized cost* as evaluation metrics, which we refer to as "reward" and "cost" for clarity and brevity. We set a uniform cost limit of 10 for all tasks.

Baselines. We compare our approach against five representative baselines: **(1) BCQL** (Fujimoto et al., 2019): A batch-constrained Q-learning with an adaptive Lagrangian penalty on constraint violations. **(2) CPQ** (Xu et al., 2022): A Q-learning methods that penalize unsafe and out-of-distribution state–action pairs. **(3) CDT** (Liu et al., 2023b): A transformer-based offline safe RL method that learns cost-conditioned action generators for constraint enforcement. **(4) LSPC** (Koirala et al., 2024): A latent safety-constrained approach that uses a conditional variational autoencoder to model safety in the latent space. **(5) FISOR** (Zheng et al., 2024): A feasibility-guided method that uses a diffusion model to policy sampling.

Main Results Table 1 summarizes results on Safety-Gymnasium, Bullet-Safety-Gym, and Safe MetaDrive. Overall, our method learns safe policies with competitive returns. BCQL uses a Lagrangian trade-off but often fails to meet safety constraints; CPQ is more conservative and im-



(a) Cost critic change (b) Value critic change (c) Decoder log probability

Figure 2: Example visualization of the refiner principle on `CarRun`. The refiner order is fixed to be safety-reward-shared. Each refiner applies small residual updates to the sampled action, steering it toward safety, high reward, and data-supported regions. These objectives are often in tension, so the resulting optimization trajectories can be contrasting.

proves safety at the cost of reward; and CDT, though capable of high returns via target conditioning, tends to violate safety more frequently. FISOR and LSPC are strong baselines with distinct characteristics. FISOR produces uniformly safe but slightly conservative policies via feasibility guidance, while LSPC is more aggressive—seeking the most rewarding action in a learned safe latent space—which can become unreliable under OOD states/actions. Our FLRP trains safety and shared refiners to concentrate probability mass in high-density regions of the encoder's latent space, naturally biasing actions toward on-support, safer choices. FLRP performs strongly on Safety-Gymnasium and Bullet-Safety-Gym, and is mildly conservative on Safe MetaDrive due to limited overlap between high-reward and low-cost regions, which complicates hard-constrained optimization. Even so, it enforces safety effectively, achieving violation rates far below the second-best method (e.g., 0.18 vs. 0.40 in Safety-Gymnasium, 0.04 vs. 0.88 in Bullet-Safety-Gym, and 0.19 vs. 0.38 in Safe MetaDrive) while maintaining strong performance.

5 Ablation Study and Analysis

Justification of Each Refiner. A core challenge in safe RL is reconciling reward maximization with safety constraints, which can pull updates in opposite directions. We illustrate this with an example from the `CarRun` task (Figure 2). In this particular state, the regions associated with high reward and high safety are largely non-overlapping. Additionally, these regions may not align well with the high-density area of the action decoder. As a result, the refinement steps taken by the reward expert and safety expert can diverge significantly, often steering the latent action representation into areas that are poorly supported by the decoder—potentially leading to OOD issues. A shared refiner can stabilize and regularize this process by keeping actions on support while balancing both experts.

HJ-feasibility Function. We first assess the benefit of incorporating HJ reachability. We replace the feasibility function with a cost value function and declare states/actions feasible if their cost is below the empirical 75th percentile of the zero-violation samples; we denote this variant as *w/o HJ*. As reported in Table 2, this heuristic thresholding yields noisier feasibility estimates, which in turn leads to higher evaluation costs and lower returns than the HJ-based approach. In contrast, HJ reachability propagates safety constraints through the dynamics, which is robust to sampling noise and uneven cost distributions. The results indicate that structured HJ reachability is crucial for stable constraint satisfaction in offline settings.

Table 2: Ablations on HJ reachability.

Task	w/o HJ		FLRP	
	$r\uparrow$	$c\downarrow$	$r\uparrow$	$c\downarrow$
AntRun	0.65	0.13	0.52	0.00
BallRun	0.08	0.14	0.16	0.00
CarRun	0.83	0.13	0.87	0.00
DroneRun	0.16	5.24	0.59	0.02
AntCircle	0.23	0.01	0.45	0.25
BallCircle	0.44	0.00	0.46	0.00
CarCircle	0.63	0.49	0.66	0.06
DroneCircle	0.56	0.67	0.54	0.00

Other Ablations. We further examine the effect of the prior. As a comparison, we train a variant that replaces our flow-based prior with a conventional Gaussian prior and report results in Table 3. The flow prior consistently yields higher returns and lower costs.

Table 3: Ablations on the prior used.

Task	Gaussian Prior		Flow Prior	
	$r \uparrow$	$c \downarrow$	$r \uparrow$	$c \downarrow$
CarButton1	-0.14	0.22	0.03	0.36
CarButton2	0.01	0.82	0.04	0.38
CarPush1	0.07	0.08	0.20	0.04
CarPush2	0.06	0.00	0.24	0.36
CarGoal1	0.06	0.00	0.27	0.00
CarGoal2	0.05	0.74	0.20	0.28

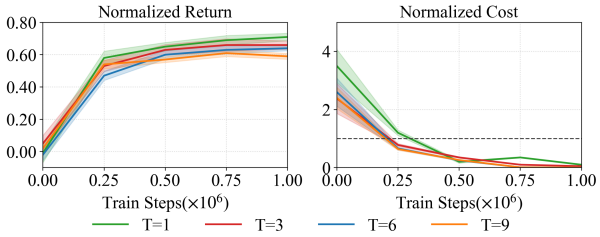


Figure 3: Ablation on the number of refinement steps.

We also study the number of refinement steps T at inference on `CarCircle`. We do not vary the refinement order: the safety expert is always applied first, and the shared expert last. This design choice reflects our latent geometry—density concentrates on safety rather than reward—so an early safety refinement helps place trajectories in high-density (feasible) regions. The intermediate refiners alternate between safety and reward experts. As shown in Figure 3, increasing T reduces cost and variability: a larger T is more likely to explore the learned latent space and lowers the rate of out-of-distribution actions. The trade-off is that a very large T can induce slightly more conservative behavior. In practice, an intermediate value (e.g., $T = 3$) can yield a favorable trade-off.

6 Related Work

Offline safe RL aims to learn constraint-satisfying policies purely from fixed datasets, avoiding risky online interaction. A first line of work imposes *penalization* or *Lagrangian* terms on value learning to satisfy cost requirements—e.g., CPQ (Xu et al., 2022), BCQ-Lag (Fujimoto et al., 2019), and BEAR-Lag (Liu et al., 2023a)—which explicitly penalize constraint violations in the Bellman targets or objectives. A complementary direction uses distribution correction and importance weighting: COptiDICE (Lee et al., 2022) models the stationary distribution and optimizes the policy under constraints directly, and is regarded as a representative approach for constrained offline RL. Sequence-modeling-based methodslike CDT (Liu et al., 2023b) and SaFormer (Zhang et al., 2023) incorporate safety into Decision Transformer (Chen et al., 2021) and achieve target-conditioned policy. Generative approaches further expand this thread: VOCE (Guan et al., 2023) reframes OSRL as probabilistic inference with a conservative variational objective, while OASIS (Yao et al., 2024) reshapes data distributions via a conditional diffusion model (Ho et al., 2020) to facilitate safe policy learning from imperfect logs. However, most of the above primarily address soft constraints. To mitigate this, recent studies (Ganai et al., 2023; Yu et al., 2022; Wang et al., 2023) adopt concepts from safe control, such as control barrier functions (CBF) (Ames et al., 2019) and Hamilton–Jacobi reachability (Bansal et al., 2017), as external safety shields or supervisory signals during offline training. Closest to our work are LSPC (Koirala et al., 2024), which jointly optimizes reward and cost in a latent space with a CVAE-based safety prior, and FISOR (Zheng et al., 2024), which couples diffusion-policy learning and HJ-based feasibility guidance to approach zero violations.

7 Conclusion

We present a safe offline RL framework based on latent refinement. A multi-expert policy iteratively improves a base latent with structured residuals guided by separate safety and reward signals. A normalizing-flow prior shapes a feasible latent manifold, enabling precise control of safety objectives while preserving return. We prove order-agnostic bounds on the final policy distribution and show competitive or superior results on three standard safe RL benchmarks.

The main limitations arise from the feasibility critics. The offline feasibility critics use a Hamilton–Jacobi–style Bellman operator with sparse cost, which can over-conservatively estimate value; genuinely safe but rare samples may be undervalued, introducing bias or sample inefficiency. Latent-space refinement also adds hyperparameters (e.g., expert loss weights and prior shaping temperature). That said, we used a single configuration across 26 tasks, suggesting reasonable robustness. Future work includes adaptive refinement schedules, more principled objectives for shaping the prior, and hierarchical expert architectures to improve flexibility and generalization.

Ethics Statement. This research does not involve human subjects, sensitive data, or practices that pose foreseeable harm. Our methodology builds upon well-established safe offline reinforcement learning benchmarks and standard datasets that are publicly available. All experiments were conducted in simulation environments with no real-world deployment or safety risk. We have made efforts to ensure transparency and reproducibility by providing code and detailed algorithmic descriptions. We adhere to the ICLR Code of Ethics, and this work upholds responsible stewardship and scientific integrity throughout.

Reproducibility Statement. We have taken several steps to ensure the reproducibility of our work. All theoretical results, including key lemmas and corollaries, are presented with complete assumptions and detailed proofs in the appendix. Additional implementation details, including dataset setup, training pipeline, and evaluation protocol, are also provided in the appendix. We also include an anonymous link to our core source code at: `https://anonymous.4open.science/r/FLRP-9776/`

REFERENCES

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

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pp. 3420–3431. Ieee, 2019.

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. IEEE, 2017.

Normand J Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*, 2011.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Imre Csiszár and Paul C Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.

Yuhao Ding and Javad Lavaei. Provably efficient primal-dual reinforcement learning for cmdps with non-stationary objectives and constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7396–7404, 2023.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36:69764–69797, 2023.

Sven Gronauer. Bullet-safety-gym: A framework for constrained reinforcement learning. Technical report, mediaTUM, 2022.

Jiayi Guan, Guang Chen, Jiaming Ji, Long Yang, Zhijun Li, et al. Voce: Variational optimization with conservative estimation for offline safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36:33758–33780, 2023.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Vindula Jayawardana, Sirui Li, Yashar Farid, and Cathy Wu. Multi-residual mixture of experts learning for cooperative control in multi-vehicle systems. *arXiv preprint arXiv:2507.09836*, 2025.

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

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43 (11):3964–3979, 2020.

Prajwal Koirala, Zhanhong Jiang, Soumik Sarkar, and Cody Fleming. Latent safety-constrained policy approach for safe offline reinforcement learning. *arXiv preprint arXiv:2412.08794*, 2024.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Ankita Kushwaha, Kiran Ravish, Preeti Lamba, and Pawan Kumar. A survey of safe reinforcement learning and constrained mdps: A technical survey on single-agent and multi-agent safety. *arXiv preprint arXiv:2505.17342*, 2025.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

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. *arXiv preprint arXiv:2204.08957*, 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.

Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, et al. Datasets and benchmarks for offline safe reinforcement learning. *arXiv preprint arXiv:2306.09303*, 2023a.

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. PMLR, 2023b.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Johan Obando-Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob Foerster, Gintare Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep rl. *arXiv preprint arXiv:2402.08609*, 2024.

Felix Otto and Cédric Villani. Generalization of an inequality by talagrand and links with the logarithmic sobolev inequality. *Journal of Functional Analysis*, 173(2):361–400, 2000.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL `http://jmlr.org/papers/v22/19-1028.html`.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Quan Vuong, Aviral Kumar, Sergey Levine, and Yevgen Chebotar. Dasco: Dual-generator adversarial support constrained offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:38937–38949, 2022.

Xiangwei Wang, Peng Wang, Renke Huang, Xiuli Zhu, Javier Arroyo, and Ning Li. Safe deep reinforcement learning for building energy management. *Applied Energy*, 377:124328, 2025.

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

Jingda Wu, Chao Huang, Hailong Huang, Chen Lv, Yuntong Wang, and Fei-Yue Wang. Recent advances in reinforcement learning-based autonomous driving behavior planning: A survey. *Transportation Research Part C: Emerging Technologies*, 164:104654, 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.

Shoukai Xu, Mingkui Tan, Liu Liu, Zhong Zhang, Peilin Zhao, et al. Test-time adapted reinforcement learning with action entropy regularization. In *Forty-second International Conference on Machine Learning*, 2025.

Yihang Yao, Zhepeng Cen, Wenhao Ding, Haohong Lin, Shiqi Liu, Tingnan Zhang, Wenhao Yu, and Ding Zhao. Oasis: Conditional distribution shaping for offline safe reinforcement learning. *Advances in Neural Information Processing Systems*, 37:78451–78478, 2024.

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

Peipei Yu, Hongcai Zhang, Yonghua Song, Zhenyi Wang, Huiyu Dong, and Liang Ji. Safe reinforcement learning for power system control: A review. *Renewable and Sustainable Energy Reviews*, 223:116022, 2025.

Qin Zhang, Linrui Zhang, Haoran Xu, Li Shen, Bowen Wang, Yongzhe Chang, Xueqian Wang, Bo Yuan, and Dacheng Tao. Saformer: A conditional sequence modeling approach to offline safe reinforcement learning. *arXiv preprint arXiv:2301.12203*, 2023.

Ziqian Zhang, Haojie Li, Tiantian Chen, NN Sze, Wenzhang Yang, Yihao Zhang, and Gang Ren. Decision-making of autonomous vehicles in interactions with jaywalkers: A risk-aware deep reinforcement learning approach. *Accident Analysis & Prevention*, 210:107843, 2025.

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. *arXiv preprint arXiv:2401.10700*, 2024.

A LLM USAGE

The authors used large language models (LLMs), specifically ChatGPT (GPT-4), solely as a language editing assistant. The LLM was employed only for grammar correction, stylistic improvements, and minor clarity revisions of the authors' own writing.

All ideas, algorithms, experimental designs, theoretical proofs, and scientific contributions presented in this paper are the sole work of the authors. The authors take full responsibility for the technical content and claims made in the paper. No content was generated or suggested by the LLM regarding methodology, experiments, or results.

B THEORETICAL ANALYSIS

In this section, we provide the missing proofs for the theoretical results to support or validate the proposed method.

B.1 PROOF OF DEFINITION 2.

For a fixed $\gamma \in (0,1)$ and we define $V_i(s) := \min_a Q_i(s,a)$ for $i \in \{1,2\}$. Then for any (s,a),

$$\begin{aligned} \big|(\mathcal{P}^\star Q_1)(s,a) - (\mathcal{P}^\star Q_2)(s,a)\big| &= \gamma \left|\mathbb{E}_{s'}\big[\max\{h(s), V_1(s')\} - \max\{h(s), V_2(s')\}\big]\right| \\ &\le \gamma\, \mathbb{E}_{s'}\big|V_1(s') - V_2(s')\big|. \end{aligned} \tag{23}$$

Since $V_i(s') = \min_{a'} Q_i(s',a')$ and the pointwise min is 1-Lipschitz, $\big|V_1(s') - V_2(s')\big| \le \sup_{a'} |Q_1(s',a') - Q_2(s',a')| \le \|Q_1 - Q_2\|_\infty$. Taking the supremum over (s,a) yields

$$\|\mathcal{P}^\star Q_1 - \mathcal{P}^\star Q_2\|_\infty \;\le\; \gamma\, \|Q_1 - Q_2\|_\infty, \tag{24}$$

so $\mathcal{P}^\star$ is a γ-contraction under the sup norm. By Banach's fixed-point theorem, there exists a unique fixed point $Q^\star_{h,\gamma}$ and we set $V^\star_{h,\gamma}(s) := \min_a Q^\star_{h,\gamma}(s,a)$.

To connect to the undiscounted HJ-style values, assume h is bounded. Let $\gamma_n \uparrow 1$ and consider the fixed points $Q^\star_{h,\gamma_n}$. Because $\{Q^\star_{h,\gamma_n}\}_n$ is uniformly bounded and $\mathcal{P}^\star$ is continuous in γ, any limit point $Q^\dagger$ satisfies, for all (s,a),

$$\begin{aligned} Q^\dagger(s,a) \;&=\; \lim_{n\to\infty} \Big[(1-\gamma_n)h(s) + \gamma_n\, \mathbb{E}_{s'}\big[\max\{h(s), \min_{a'} Q^\star_{h,\gamma_n}(s',a')\}\big]\Big] \\ &= \mathbb{E}_{s'}\big[\max\{h(s), \min_{a'} Q^\dagger(s',a')\}\big]. \end{aligned} \tag{25}$$

This is the dynamic programming equation for the HJ-style (statewise zero-violation) feasibility values; hence $Q^\dagger = Q^\star_h$ and $V^\dagger = \min_a Q^\dagger(\cdot,a) = V^\star_h$. Therefore $Q^\star_{h,\gamma} \to Q^\star_h$ and $V^\star_{h,\gamma} \to V^\star_h$ as $\gamma \uparrow 1$. □

B.2 PROOF OF LEMMA 1.

Recall the weighted objective in full form:

$$\begin{aligned} \mathcal{L}_{\text{flow}} = \mathbb{E}_{(s,a)\sim p_\mathcal{D}}\Big[& w(s,a)\; \mathbb{E}_{z\sim q_\psi(z|s,a)}\big[-\log \pi_\theta(a \mid s,z)\big]\Big] \\ &+\; \beta\, \mathbb{E}_{(s,a)\sim p_\mathcal{D}}\Big[w(s,a)\; D_{\mathrm{KL}}\big(q_\psi(\cdot \mid s,a)\,\big\|\, p_\phi(\cdot \mid s)\big)\Big]. \end{aligned} \tag{26}$$

and we define the behavior-weighted data distribution $\tilde{p}_\mathcal{D}(s,a) := w(s,a)\, p_\mathcal{D}(s,a)/Z$ with normalizer $Z = \mathbb{E}_{p_\mathcal{D}}[w(s,a)]$ (a constant independent of (ϕ,ψ,θ)). For clarity, first consider $\beta = 1$; we return to $\beta \neq 1$ at the end. Then, up to the positive constant factor Z,

$$\mathcal{L}_{\text{flow}} = Z \cdot \mathbb{E}_{(s,a)\sim\tilde{p}_\mathcal{D}}\Big\{\mathbb{E}_{z\sim q_\psi}\big[-\log \pi_\theta(a \mid s,z)\big] + D_{\mathrm{KL}}\big(q_\psi \,\|\, p_\phi\big)\Big\}.$$

Expand the KL term inside the expectation:

$$\begin{aligned} &\mathbb{E}_{z\sim q_\psi}\big[-\log \pi_\theta(a \mid s,z)\big] + \mathbb{E}_{z\sim q_\psi}\big[\log q_\psi(z \mid s,a) - \log p_\phi(z \mid s)\big] \\ &\qquad = \mathbb{E}_{z\sim q_\psi}\Big[\log \frac{q_\psi(z \mid s,a)}{p_\phi(z \mid s)\, \pi_\theta(a \mid s,z)}\Big]. \end{aligned} \tag{27}$$

Taking the expectation over $(s,a) \sim \tilde{p}_{\mathcal{D}}$ yields

$$\begin{aligned}\frac{1}{Z}\,\mathcal{L}_{\text{flow}} &= \mathbb{E}_{(s,a)\sim\tilde{p}_{\mathcal{D}}}\mathbb{E}_{z\sim q_\psi}\left[\log \frac{\tilde{p}_{\mathcal{D}}(s,a)\,q_\psi(z \mid s,a)}{\tilde{p}_{\mathcal{D}}(s,a)\,p_\phi(z \mid s)\,\pi_\theta(a \mid s,z)}\right] \\ &= D_{\mathrm{KL}}\Big(\tilde{p}_{\mathcal{D}}(s,a)\,q_\psi(z \mid s,a) \,\Big\|\, \tilde{p}_{\mathcal{D}}(s,a)\,p_\phi(z \mid s)\,\pi_\theta(a \mid s,z)\Big).\end{aligned} \tag{28}$$

Finally, use the identity $D_{\mathrm{KL}}(P\|C\cdot Q) = D_{\mathrm{KL}}(P\|Q) - \mathbb{E}_P[\log C]$ for a positive constant density factor C that does not depend on the model parameters (ϕ,ψ,θ); here $C = \tilde{p}_{\mathcal{D}}(s,a)$. Therefore,

$$\mathcal{L}_{\text{flow}} = \text{const} \;+\; D_{\mathrm{KL}}\Big(\tilde{p}_{\mathcal{D}}(s,a)\,q_\psi(z \mid s,a) \,\Big\|\, p_\phi(z \mid s)\,\pi_\theta(a \mid s,z)\Big), \tag{29}$$

where the constant depends only on $\tilde{p}_{\mathcal{D}}$ (hence on w and the dataset) and not on (ϕ,ψ,θ). This proves the claim for $\beta = 1$.

Extension to $\beta \neq 1$. For a general $\beta > 0$, the same algebra shows that

$$\mathcal{L}_{\text{flow}} = \text{const} \;+\; D_{\mathrm{KL}}\Big(\tilde{p}_{\mathcal{D}}(s,a)\,q_\psi(z \mid s,a) \,\Big\|\, p_\phi^{(\beta)}(z \mid s)\,\pi_\theta(a \mid s,z)\Big), \tag{30}$$

with a *temperature-adjusted prior* $p_\phi^{(\beta)}(z \mid s) \propto p_\phi(z \mid s)^\beta$ (i.e., the energy scaled by β). Equivalently, if one wishes to keep p_ϕ unchanged, absorb β by rescaling the KL term or by introducing a decoder temperature; both formulations are strictly equivalent up to a parameter-independent constant. □

B.3 Proof of Lemma 2

Let $p := \Pi_\theta(\cdot|s),\ r := \pi_0(\cdot|s),\ q := \pi_\beta(\cdot|s)$ w.r.t. a common dominating measure. By the elementary inequality (chain rule with a bounded density ratio)

$$D_{\mathrm{KL}}(p\|q) = D_{\mathrm{KL}}(p\|r) + \mathbb{E}_p\big[\log \tfrac{r}{q}\big] \le D_{\mathrm{KL}}(p\|r) + \log \sup_a \frac{r(a)}{q(a)} = D_{\mathrm{KL}}(p\|r) + \log R_\theta(s). \tag{31}$$

Under a frozen decoder $T_s : \mathcal{U} \to \mathcal{A}$, we treat the transformation from base latent u to action a as a measurable pushforward mapping. Let q_u be the refined base distribution and $\mathcal{N}$ the standard Gaussian. Then the induced action distributions satisfy

$$D_{\mathrm{KL}}(T_{s\#}q_u \,\|\, T_{s\#}\mathcal{N}) \;\le\; \mathrm{KL}(q_u \,\|\, \mathcal{N}), \tag{32}$$

by the data-processing inequality (DPI) for Kullback–Leibler divergence under measurable maps; (e.g.,see Csiszár & Shields (2004)). This result states that any deterministic or stochastic channel (here, the frozen decoder T_s) cannot increase KL divergence. □

B.4 Proof of Lemma 3

Let $f_\phi : \mathbb{R}^d \to \mathbb{R}^d$ be a smooth bijection (the prior flow). Define $q_z = f_{\phi\#}q_u$ and $p_\phi = f_{\phi\#}\mathcal{N}$. By the change-of-variables formula,

$$q_z(z) \;=\; q_u(u)\left|\det \frac{\partial u}{\partial z}\right|, \qquad p_\phi(z) \;=\; \mathcal{N}(u)\left|\det \frac{\partial u}{\partial z}\right|, \quad z = f_\phi(u). \tag{33}$$

Hence

$$D_{\mathrm{KL}}(q_z\|p_\phi) = \int q_z(z)\log\frac{q_z(z)}{p_\phi(z)}\,\mathrm{d}z = \int q_u(u)\log\frac{q_u(u)}{\mathcal{N}(u)}\,\mathrm{d}u = D_{\mathrm{KL}}(q_u\|\mathcal{N}), \tag{34}$$

i.e., KL is invariant under the bijection f_ϕ.

Let $T_s : \mathbb{R}^d \to \mathcal{A}$ be the deterministic decoder mapping (e.g., decoded mean) with frozen θ. The data-processing inequality for f-divergences (including KL) under a measurable pushforward gives

$$D_{\mathrm{KL}}\big(T_{s\#}q_z \,\big\|\, T_{s\#}p_\phi\big) \;\le\; D_{\mathrm{KL}}(q_z\|p_\phi). \tag{35}$$

With $\pi = T_{s\#}q_u = T_{s\#}q_z$ and $\pi_0 = T_{s\#}\mathcal{N} = T_{s\#}p_\phi$, we obtain $D_{\mathrm{KL}}(\pi\|\pi_0) \le \mathrm{KL}(q_z\|p_\phi) = D_{\mathrm{KL}}(q_u\|\mathcal{N})$, which proves Eq. 18.

B.5 Proof of Corollary 1

For the Wasserstein bound, write $\pi = T_{s\#}q_z$ and $\pi_0 = T_{s\#}p_\phi$ with $T_s = g_\theta(\cdot, s)$. If g_θ is L_g-Lipschitz on the latent chart, then the pushforward is L_g-Lipschitz in W_2:

$$W_2(\pi, \pi_0) \;\le\; L_g\, W_2(q_z, p_\phi). \tag{36}$$

By Talagrand's T_2 inequality (Gaussian reference or log-Sobolev under mild conditions) (Otto & Villani, 2000), $W_2(q_z, p_\phi) \le \sqrt{2\, D_{\mathrm{KL}}(q_z \| p_\phi)}$, and Lemma 3 implies $W_2(\pi, \pi_0) \le L_g \sqrt{2\, D_{\mathrm{KL}}(q_u \| \mathcal{N})}$.

For total variation (TV) and OOD probability, the triangle inequality yields $\mathrm{TV}(\pi, \pi_\beta) \le \mathrm{TV}(\pi, \pi_0) + \mathrm{TV}(\pi_0, \pi_\beta)$. Pinsker's inequality (Csiszár & Shields, 2004) gives $\mathrm{TV}(\pi, \pi_0) \le \sqrt{\frac{1}{2}\, D_{\mathrm{KL}}(\pi \| \pi_0)} \le \sqrt{\frac{1}{2}\, D_{\mathrm{KL}}(q_u \| \mathcal{N})}$, using Lemma 3. For any measurable $\mathcal{O}$,

$$\pi(\mathcal{O}) - \pi_\beta(\mathcal{O}) \;\le\; \mathrm{TV}(\pi, \pi_\beta) \;\le\; \sqrt{\tfrac{1}{2}\, D_{\mathrm{KL}}(q_u \| \mathcal{N})} + \mathrm{TV}(\pi_0, \pi_\beta). \tag{37}$$

Rearranging completes the proof.

Remark. The Wasserstein bound in Corollary 1 relies on the Lipschitz continuity of the decoder g_θ with constant L_g. We note that this is a mild and practically enforceable assumption. During training, the decoder's Lipschitz constant can be implicitly constrained through techniques such as weight normalization Salimans & Kingma (2016), spectral normalization Miyato et al. (2018), or the gradient penalty Gulrajani et al. (2017), which are commonly used in generative modelling to enhance stability and generalization. Consequently, the theoretical bounds derived herein are not only sound but also practically relevant, as the key quantity $D_{\mathrm{KL}}(q_u|\mathcal{N})$ remains the primary lever for controlling distributional shift.

B.6 Order-agnostic bounds for sequential refinement

We formalize that the KL/Wasserstein/TV bounds in Lemma 3 and Corollary 1 are independent of the update order used by the experts.

Proposition 1 (Order-agnosticity of base-space bounds)**.** *Let R be any (possibly stochastic) measurable refinement operator on the base space that maps the standard Gaussian $\mathcal{N}$ to a refined distribution $q_u = R(\mathcal{N})$, obtained by any composition/order of expert updates (e.g., Gauss–Seidel, Jacobi, interleaved mini-steps) subject to a trust region $\|u_T - u_0\| \le \rho$. With the prior flow f_ϕ and decoder g_θ fixed (as in Sec. 3.2), define $\pi = T_{s\#}q_u$ and $\pi_0 = T_{s\#}\mathcal{N}$ where $T_s(u) = g_\theta(f_\phi(u; s), s)$. Then the conclusions of Lemma 3 and Corollary 1 hold verbatim with this q_u:*

$$\begin{aligned} D_{\mathrm{KL}}\big(\pi(\cdot \mid s) \,\big\|\, \pi_0(\cdot \mid s)\big) \;&\le\; D_{\mathrm{KL}}\big(q_u(\cdot \mid s) \,\big\|\, \mathcal{N}\big), \\ W_2\big(\pi(\cdot \mid s), \pi_0(\cdot \mid s)\big) \;&\le\; L_g \sqrt{2\, D_{\mathrm{KL}}(q_u \| \mathcal{N})}, \end{aligned} \tag{38}$$

and the TV/Pinsker OOD bound remains unchanged.

Proof. The proofs of Lemma 3 and Corollary 1 use only: (i) invariance of KL under the bijection f_ϕ; (ii) data-processing for pushforwards through the frozen decoder g_θ; (iii) Talagrand/Pinsker inequalities. None of these depend on the *path* that produces q_u, only on the *resulting* distribution q_u. Any expert ordering defines a measurable map whose pushforward of $\mathcal{N}$ is q_u; substituting this q_u into the same steps yields the stated bounds. The optional trust region ensures KL finiteness and well-definedness but does not affect order independence. □

C Implementation Details

In this section, we describe our experimental framework and implementation of the proposed method, including benchmark and datasets, task descriptions and evaluation metrics, and training details.

C.1 Benchmark Details

We use the Datasets for Safe Reinforcement Learning (DSRL) benchmark suite (Liu et al., 2023a) to train and evaluate our method as well as all baselines. DSRL provides 38 offline datasets spanning multiple safe RL environments (Safety-Gymnasium, Bullet-Safety-Gym, and Safe MetaDrive) with varying difficulty levels. These datasets follow a D4RL-style (Fu et al., 2020) API and include detailed cost signals in addition to reward returns.

For the baselines, we adopt the authors' official implementations and default hyperparameters when available (especially for FISOR and LSPC). For other methods (BCQL / BCQ-Lag, CPQ, CDT), we use the OSRL framework's implementations and settings to ensure fair comparison.

C.2 Task Descriptions

Below are the three environment suites used in our experiments, with their main task types and distinguishing safety vs. complexity features. Figure 4 shows three example visualizations.

C.2.1 Safety-Gymnasium

Safety-Gymnasium (Ji et al., 2023) is a unified MuJoCo-based benchmark collection offering a variety of continuous control tasks (e.g. Goal, Button, Push, Circle, Velocity, etc.). Agents include Point, Car, Ant, HalfCheetah, etc. The tasks vary both in goal structure (e.g. reach a goal, push an object, navigate through buttons) and safety constraints (velocity limits, obstacle avoidance, collision cost). Some tasks include hazards or "sigwalls" that act as soft or hard boundaries. These tasks test both navigation and locomotion under safety constraints.

C.2.2 Bullet-Safety-Gym

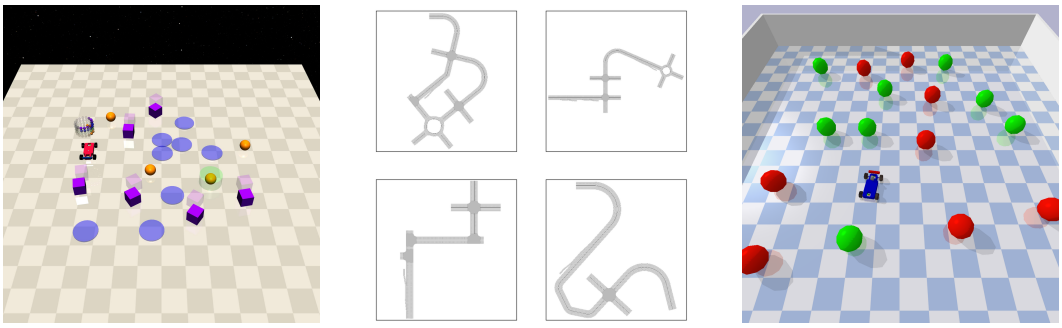
Bullet-Safety-Gym (Gronauer, 2022) is based on the PyBullet physics engine. It includes similar task types (Circle, Run, Gather, Reach) with agents such as Ball, Car, Drone, Ant. The dynamics tend to have shorter horizons and more variability in physics (collision, friction) compared to MuJoCo, which raises safety risk under state/action noise. Cost signals usually arise from collisions or from exceeding safe boundaries. This makes the tasks more challenging in terms of generalization and handling unsafe transitions.

C.2.3 Safe MetaDrive

MetaDrives (Li et al., 2022) is a simulator for driving/traffic/autonomous vehicle tasks under safety constraint. Its "safe RL" subset includes tasks with realistic road networks, dynamic agents, procedural map generation, traffic rules, and hazards. Observations often include vehicle state, road context; actions are continuous control of speed/steering. Safety constraints include collisions, lane infractions, and staying within road limits. These tasks are more realistic in terms of environment unpredictability, driving constraints, and possibly partial observability or environmental stochasticity.

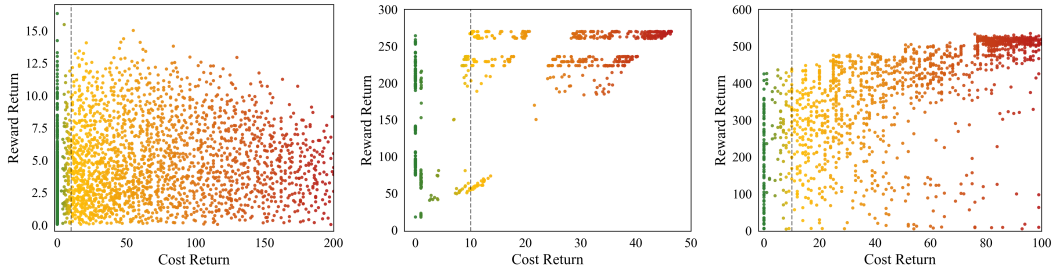
C.3 Dataset Visualization

We further present the distribution of offline trajectories in the cost–return space across three representative environments, as shown in Figure 5. In the `CarPush` task from Safety-Gymnasium, the reward distribution is narrow and low, while the cost spans a wide range. This results in a weak correlation between reward and safety: most trajectories incur significant costs even when achieving only modest returns, making strict constraint satisfaction particularly challenging. In the `MediumMean` task from Safe MetaDrive, the reward exhibits distinct discrete bands, each associated with a specific cost level. This reflects mode-switching behaviors and a strong reward–cost coupling; although feasible trajectories exist, achieving high reward under tight cost limits requires careful selection among these behavioral clusters. The `CarRun` task from Bullet Safety Gym demonstrates a smoother trade-off frontier, where reward gradually increases with cost, forming a continuous and diverse distribution. While safe, high-reward trajectories remain sparse, the presence of mid-reward, intermediate-cost episodes renders this dataset more amenable to constrained policy optimization compared to the other two.



(a) Safety Gymnasium (b) Metadrive (c) Bullet Safety Gym

Figure 4: Example visualization from the simulation environments used in our experiments.



(a) CarPushGymnasium (b) MediumMean (c) BulletCarRun

Figure 5: Example visualization of the dataset used in our experiments.

C.4 Evaluation Metrics

We evaluate the performance of all methods using two metrics: *normalized reward return* and *normalized cost return*, following standard evaluation practices used in offline RL benchmarks like D4RL Fu et al. (2020) and adopted by recent safe RL methods such as CDT Liu et al. (2023b), LSPC (Koirala et al., 2024) and FISOR (Zheng et al., 2024). The normalized reward is defined as:

$$R_{\text{norm}} = \frac{R_\pi - r_{\min}(\mathcal{T})}{r_{\max}(\mathcal{T}) - r_{\min}(\mathcal{T})} \tag{39}$$

where R_π is the total reward return of the trained policy π, and $r_{\max}(\mathcal{T})$, $r_{\min}(\mathcal{T})$ denote the maximum and minimum reward returns observed in the dataset $\mathcal{T}$, respectively.

The normalized cost is computed as:

$$C_{\text{norm}} = \frac{C_\pi}{\kappa + \epsilon} \tag{40}$$

where C_π is the total cost return of policy π, κ is the cost limit, which we set to 10 for all tasks, and ϵ is a small constant added to avoid numerical instability when $\kappa = 0$.

C.5 Training Details

For all baseline methods, we adopt their default hyperparameter configurations. To ensure a fair comparison across all methods, we set the rollout length for each task to match the maximum number of allowed interaction steps. The cost limit for the baselines is set to 10 for all tasks. The common key hyperparameters used for our method and baselines are shown in Table 4. Table 5 lists other key hyperparameters used for FLRP. We apply the same configuration across all tasks and environments without per-task tuning.

The pseudocode for FLRP is provided in Algorithm 1. All experiments were conducted on eight NVIDIA RTX 6000 Ada Generation GPUs, each with 48 GB of memory. Each experiment is run with 3 random seeds, and results are averaged over 10 evaluation episodes per seed.

Table 4: Model Configuration Parameters

Parameter	CPQ	BCQ-L	CDT	LSPC	FISOR	FLRP
Common Settings:						
Training steps			1×10^6			
Batch size			512			
Discount factor			0.99			
Activate function			ReLu			
Algorithm-Specific Settings:						
Hidden layer size	256	256	256	256	256	256
Soft update rate (τ)	0.005	0.005	0.005	0.005	0.001	0.001
Cost limit	10	10	10	–	–	–
Learning Rates ($\times 10^{-3}$):						
Actor learning rate	1.0	1.0	0.1	0.3	0.3	0.3
Critic learning rate	1.0	1.0	0.1	0.3	0.3	0.3

Table 5: Hyperparameters of FLRP.

Parameter	Value
Expectile τ	0.9
Asymmetric L2 loss coeff	0.9
Target temperature	3
Value temperature	5
Advantage weight clip (reward)	$(-\infty,\ 100]$
Advantage weight clip (cost)	$(-\infty,\ 150]$
Refine steps T	3
Refiner loss weight $\lambda_r, \lambda_h, \lambda_{sh}$	1,1,0.5

Algorithm 1 FLRP Training

Require: Offline dataset $\mathcal{D}$

1: **Init** critics $(Q_r, V_r), (Q_h, V_h)$; flows p_ϕ, q_ψ; decoder π_θ; refiners $\{\mathcal{R}_\text{s}, \mathcal{R}_\text{r}, \mathcal{R}_\text{sh}\}$
2: **while** not converged **do**
3: Sample minibatch $(s, a) \sim \mathcal{D}$; draw $z \sim q_\psi(z \mid s, a)$
4: Update safety critics (Q_h, V_h) by the HJ-style backup Eq. 8 and Eq. 9
5: Update reward critics (Q_r, V_r) by offline TD/advantage target Eq. 21 and Eq. 22
6: Update the flows using the flow objective Eq. 13
7: Training the refiners using the refiner objective Eq. 17
8: **end while**