Safe Offline Reinforcement Learning via Chance-Constrained Policy Filtering

Kangyeon Kim, Seonho Yoo, and Heejin Ahn

***Abstract*— This paper proposes a chance-constrained policy filtering method for safe offline reinforcement learning (RL) designed to limit the probability of safety violation to a user-specified threshold. The proposed approach constructs a safety-filtered policy that accepts or rejects candidate actions based on a learned cost estimator. To support satisfaction of the target safety constraint, we further calibrate the estimator via posterior computation using the offline dataset. The proposed method offers significant flexibility, allowing for dynamic adjustment of constraint thresholds during deployment without additional retraining. We empirically validate our method on offline RL robot navigation tasks in the OSRL benchmark, demonstrating that it meets user-specified thresholds while achieving additional performance gains via end-to-end back-propagation.**

I. Introduction

Safety-critical autonomous systems, such as industrial automated guided vehicles (AGVs) and self-driving cars, require reliable decision-making mechanisms to prevent catastrophic failures. Ensuring safety in decision-making is therefore a crucial requirement in such systems. A common approach is to enforce constraints that restrict the agent from entering hazardous regions. To formalize this conceptually, the decision-making process is typically formulated as a constrained optimization problem that aims to optimize an objective function while satisfying safety requirements.

Such optimization problems are generally addressed through two main approaches: model-driven and data-driven. Rooted in control theory, model-driven approaches compute safe control actions based on a given dynamics model, such as robust model predictive control (MPC) and tube-based MPC [1]. In these approaches, constraints can incorporate various sources of uncertainty, including exogenous disturbances, modeling inaccuracies, and variations in operating conditions [2]. In contrast, data-driven approaches do not rely on explicit dynamics models and instead learn policies or value functions directly from data. This enables them to capture complex nonlinear behaviors and potentially adapt more flexibly to changing environments, particularly when the underlying model is inaccurate. Nevertheless, providing safety guarantees for data-driven approaches remains challenging due to inherent uncertainties. These data-driven models may overfit to the training data or encounter out-of-distribution inputs during real-world deployment, potentially leading to erroneous predictions or unsafe decisions. Therefore, in safety-critical decision-making, relying solely on data-driven approaches can be risky.

Reinforcement learning (RL) is a representative data-driven approach that learns a decision-making policy through interaction with the environment. Standard online RL, however, is often unsuitable for safety-critical systems, as its trial-and-error exploration can drive the agent toward hazard states [3]. To address this limitation, offline RL trains policies using pre-collected datasets without requiring online interaction. Despite avoiding unsafe exploration during training, offline RL policies still face deployment risks due to distribution shifts and limited dataset coverage [4]. Discrepancies between the static offline dataset and the dynamic real-world environment may cause the learned policy to select out-of-distribution (OOD) actions in previously unseen states, potentially leading to safety violations.

Existing methods attempt to mitigate these issues by penalizing OOD actions to induce conservatism [5], applying stationary distribution-correction techniques [6], or leveraging generative models and sequential modeling to produce constraint-aware behaviors [7], [8], [9]. While these approaches address important challenges in safe offline RL, they primarily focus on distribution shift or constraint-aware policy learning, rather than explicitly incorporating the estimation uncertainty of learned neural estimators into deployment-time safety assessment. Beyond offline RL, recent studies have considered such uncertainty in online settings, for example by combining adaptive conformal prediction with control barrier functions [10] to account for inaccuracies in learned dynamics, or by integrating conformal quantile calibration with evidential learning [11]. However, these approaches are not tailored to safe offline RL, and thus do not consider the uncertainty of estimators learned solely from offline data.

In this paper, we propose a chance-constrained policy filtering method tailored to safe offline RL. Unlike conventional offline RL algorithms that use the dataset solely for policy training, our approach additionally reuses the dataset at deployment to estimate the probability of safety violation. The proposed method employs a learned cost estimator to construct a safety-filtered policy that selectively accepts or rejects candidate actions. The key distinction of our approach is that we do not treat the learned estimator as perfectly accurate. Instead, we explicitly quantify the risk of

*This work was partly supported by the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) grant funded by the Ministry of Science and ICT of the Republic of Korea (IITP-2026-RS-2023-00259991, 50%), and by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education of the Republic of Korea.

The authors are with the School of Electrical Engineering, KAIST, Daejeon, Republic of Korea. ydk1235@naver.com, sh.yoo@kaist.ac.kr, heejin.ahn@kaist.ac.kr

Project Page: https://safeofflinerl-cpf.github.io

undetected safety violations and incorporate this uncertainty into the filtering mechanism. To ensure that estimator-based decisions satisfy a chance constraint, which allows violations only with a prescribed probability, we iteratively calibrate the estimator via posterior computation on the offline dataset. We further employ conservative testing to derive a conservative upper bound on the posterior violation probability computed from finite and potentially biased offline data. The entire procedure can be incorporated into end-to-end back-propagation, yielding modest performance gains.

Our method operates as an inference-time safety filter and can therefore be readily integrated with existing offline RL algorithms. It supports varying safety thresholds during deployment without additional training. Experimental results show that the proposed method empirically satisfies various user-specified probabilistic safety thresholds across diverse navigation tasks in the OSRL benchmark. Furthermore, our framework enhances task performance via end-to-end back-propagation while preserving safety, and achieves the highest reward return in more settings than the baselines. This work extends the AI safety ensuring framework [12] to the offline RL setting, showing that safety can be maintained at deployment using only offline data.

The main contributions are summarized as follows:

1) We propose a chance-constrained policy filtering framework for safe offline RL by reusing the offline dataset at deployment.
2) We develop a posterior-based calibration mechanism that explicitly quantifies and controls the misclassification risk of a learned cost estimator, thereby supporting chance-constraint satisfaction even when the estimator is imperfect.
3) We demonstrate the versatility of our method by integrating it with various offline RL algorithms and show that it accommodates varying safety thresholds during deployment without retraining.
4) We empirically validate our approach on offline RL robot navigation tasks, showing that it satisfies the prescribed constraints and further improves performance via end-to-end back-propagation.

II. Related Work

a) Offline Reinforcement Learning: Offline RL aims to train agents solely from pre-collected datasets without further interaction with the environment. A central challenge in this setting is the distribution shift between the offline dataset and the state–action distribution induced by the learned policy during deployment [4]. This mismatch can result in significant performance degradation or even unsafe behavior. One common strategy to mitigate distribution shift is behavior regularization, which constrains the learned policy to remain close to the behavior policy underlying the dataset, thereby reducing extrapolation and bootstrapping errors. Representative methods include BCQ [13], BEAR [14], BRAC [15], and TD3+BC [16]. Another line of work addresses distribution shift via stationary distribution correction. GenDICE [17] and DualDICE [18] estimate the density ratio between the target policy's stationary distribution and that of the dataset, while OptiDICE [19] and Diffusion-DICE [20] incorporate this ratio directly into offline policy optimization. Complementary to policy- and distribution-level approaches, conservative value learning methods, such as CQL [21], mitigate overestimation by learning a conservative lower bound of the Q-function using only offline data. Recent approaches cast offline RL as sequence modeling with transformer-based methods, such as Decision Transformer (DT) [22] and Trajectory Transformer (TT) [23], to avoid bootstrapping errors.

b) Safe Offline Reinforcement Learning: Safe offline RL aims to ensure safety during deployment while learning exclusively from pre-collected data. The central challenge arises from distribution shift and limited coverage of safety-critical tail events, where unreliable estimates of out-of-distribution (OOD) actions can lead to safety violations. To address this, existing works mainly pursue (i) Lagrangian-based methods, (ii) distribution-correction methods, (iii) generative or sequential modeling of safe behaviors, and (iv) action filtering via learned cost estimators.

Lagrangian-based methods, such as CPQ [5], incorporate safety penalties to induce conservative behavior. Distribution-correction methods, such as COptiDICE [6], estimate the stationary density ratio of the optimal safe policy relative to the dataset, enabling reward maximization under cost constraints without querying OOD actions. Generative and sequential methods, such as CDT [7], OASIS [8], and FISOR [9], mitigate OOD risks by reformulating safe offline RL as conditional generative or sequence modeling problems. Finally, action-filtering approaches reject unsafe candidates using learned cost critics; for example, CAPS [24] switches among candidate policies by filtering actions under a cost constraint. However, these methods rely on the assumption that their underlying function approximators—cost critics, density ratio estimators, generative models, and sequence predictors—are sufficiently accurate. They do not explicitly account for function approximation errors induced by finite and biased datasets. Consequently, theoretical guarantees established under idealized assumptions may fail to translate into reliable safety guarantees at deployment.

Our method employs a learned cost estimator to filter candidate actions, enabling zero-shot adaptation to varying safety thresholds at deployment. While CAPS and CDT offer similar adaptability, we additionally account for the estimation error of the learned estimator, ensuring chance-constraint satisfaction despite its inaccuracies.

III. Problem Setup

Safe decision-making is formulated as a Constrained Markov Decision Process (CMDP), which augments the standard Markov Decision Process framework with cost constraints to enforce safety. A CMDP is defined by the tuple $M = \langle S, A, T, r, c, \gamma, \mu_0 \rangle$, where S denotes the state space, A denotes the action space, $T : S \times A \times S \to [0,1]$ is the transition probability, $r : S \times A \to [r_{\min}, r_{\max}] \subset \mathbb{R}$ is the reward function, $c : S \times A \to C = \{0, 1, \ldots, c_{\max}\} \subset \mathbb{Z}$ is an integer-valued cost

function, $\gamma \in (0,1]$ is the discount factor, and $\mu_0 : S \to [0,1]$ is the initial state distribution.

At time t, the state and action are denoted by $\mathbf{s}_t \in S$ and $\mathbf{a}_t \in A$, respectively, with $\mathbf{s}_0 \sim \mu_0$ and $\mathbf{s}_{t+1} \sim T(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$. The reward and cost are $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$ and $c_t = c(\mathbf{s}_t, \mathbf{a}_t)$. For simplicity, we focus on the binary-cost setting $c_{\max} = 1$, i.e., $C = \{0,1\}$, where $c_t = 1$ denotes a safety violation and $c_t = 0$ otherwise. Nevertheless, our proposed method readily extends to the general case $c_{\max} > 1$.

Let $\pi : S \times A \to [0,1]$ denote a deployed stochastic policy, where $\pi(\mathbf{a} \mid \mathbf{s})$ defines a conditional probability distribution over actions for each state $\mathbf{s}$. The state-action distribution is defined as

$$d^{\pi}(\mathbf{s},\mathbf{a}) := \begin{cases} (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \cdot \Pr(\mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}) & \text{if } \gamma < 1, \\ \lim_{T\to\infty} \frac{1}{T+1} \sum_{t=0}^{T} \Pr(\mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}) & \text{if } \gamma = 1, \end{cases}$$

where the action is given by $\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t)$. To address safety concerns, we constrain the probability under policy π of visiting hazardous regions. Specifically, we impose the following chance constraint on a deployed policy π:

$$\mathbb{P}_{\pi}[c(\mathbf{s},\mathbf{a}) = 1] \le \delta, \tag{1}$$

where $\mathbb{P}_{\pi}[c(\mathbf{s},\mathbf{a}) = 1] := \int_{(\mathbf{s},\mathbf{a}):\, c(\mathbf{s},\mathbf{a})=1} d^{\pi}(\mathbf{s},\mathbf{a})\, d\mathbf{s}\, d\mathbf{a}$, and δ is a user-specified safety threshold.

We assume access to an offline dataset D_{off} consisting of N_τ trajectories with horizon H, $\{\tau^i\}_{i=1}^{N_\tau}$. Each trajectory is given by $\tau^i = \{(\mathbf{s}_t^i, \mathbf{a}_t^i, r_t^i, c_t^i, \mathbf{s}_{t+1}^i)\}_{t=0}^{H-1}$. Let $d^{D_{\text{off}}}(\mathbf{s},\mathbf{a})$ denote the state-action distribution induced by the dataset, and let π_{off} denote a fixed base policy learned solely from D_{off}. Given D_{off} and π_{off}, our goal is to construct a deployed policy π by applying an inference-time safety filter to π_{off}, such that π satisfies the chance constraint (1) while minimally degrading the performance of π_{off}.

IV. Methods

Fig. 1 illustrates the overall pipeline of our approach. Our method consists of three components: (i) a cost estimator that predicts whether executing action $\mathbf{a}$ at state $\mathbf{s}$ is safe (Fig. 1-**A**), (ii) a filtering rule that constructs a deployed policy π by accepting or rejecting actions based on the estimator's output (Fig. 1-**A**), and (iii) a posterior computation and calibration procedure that adjusts the estimator so that the resulting filtering rule satisfies the target safety constraint (Fig. 1-**B**). Finally, we optionally fine-tune the cost estimator via backpropagation to further improve performance (Fig. 1-**C**).

A. Cost Estimator

Since the true cost function $c(\mathbf{s},\mathbf{a})$ is unknown prior to execution and its value can only be observed after executing an action, the chance constraint (1) cannot be enforced directly at decision time. To address this, we introduce a cost estimator

$$\hat{c}(\mathbf{s},\mathbf{a}) := g\big(f_\phi(\mathbf{s},\mathbf{a})\big),$$

where $f_\phi : S \times A \to \mathbb{R}^{n_c}$ is a neural network parameterized by ϕ and $g : \mathbb{R}^{n_c} \to C$ is a post-processing function (e.g., $\arg\max$). In our binary-cost setting, the estimator performs binary classification with $n_c = 2$. We interpret $\hat{c}(\mathbf{s},\mathbf{a}) = 0$ as *predicted-safe* and $\hat{c}(\mathbf{s},\mathbf{a}) = 1$ as *predicted-unsafe*.

a) Signed Safety Margin: Although the estimator can adopt any classifier architecture, we implement it using a signed safety margin formulation. The estimator $\hat{c}$ is pre-trained using the offline dataset D_{off}. For each transition $(\mathbf{s}_t^i, \mathbf{a}_t^i, r_t^i, c_t^i, \mathbf{s}_{t+1}^i) \in D_{\text{off}}$, we use $(\mathbf{s}_t^i, \mathbf{a}_t^i)$ as input. Let $o_h(\mathbf{s}_{t+1}^i)$ denote the h-th safety-relevant component (e.g., a hazard occupancy reading from the lidar sensor) where $h \in \{1,\dots,n_h\}$. We define the *signed safety margin* as

$$z^h(\mathbf{s}_{t+1}^i) := o_h(\mathbf{s}_{t+1}^i) - m_h,$$

where m_h is a predefined safety margin, and $z^h(\mathbf{s}_{t+1}^i) > 0$ indicates a safety margin violation.

Although $\mathbf{s}_{t+1}^i$ is available offline, it is not observable at decision time. We therefore learn to predict the margin from $(\mathbf{s}_t^i, \mathbf{a}_t^i)$ and denote the prediction by $z_\phi^h(\mathbf{s}_t^i, \mathbf{a}_t^i)$. The network f_ϕ is implemented as a multi-head network that produces $\mathbf{z}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i) \in \mathbb{R}^{n_h}$. The final two-dimensional logits are constructed as

$$f_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i) = \big[0,\ \max_h z_\phi^h(\mathbf{s}_t^i, \mathbf{a}_t^i)\big]. \tag{2}$$

An action is classified as unsafe if any predicted margin is positive, i.e., $\max_h z_\phi^h(\mathbf{s}_t^i, \mathbf{a}_t^i) > 0$.

b) Training: We train the cost estimator using the loss function $L = L_{\text{reg}} + \alpha_L L_{\text{bce}}$, where α_L is a hyperparameter. The regression loss $L_{\text{reg}} = \big\|\mathbf{z}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i) - \mathbf{z}(\mathbf{s}_{t+1}^i)\big\|_2^2$ encourages accurate prediction of signed safety margins, where $\mathbf{z}(\mathbf{s}_{t+1}^i) \in \mathbb{R}^{n_h}$ denotes the signed safety margins vector. The binary classification loss $L_{\text{bce}} = \sum_{h=1}^{n_h} \text{BCEWithLogits}\Big(z_\phi^h(\mathbf{s}_t^i, \mathbf{a}_t^i), \mathbb{I}\{z^h(\mathbf{s}_{t+1}^i) > 0\}\Big)$ enforces consistency with the observed violation indicators, where $\mathbb{I}$ is the indicator function. This joint objective improves the predictive accuracy of $\hat{c}$.

B. Filtering

We use the cost estimator for deployment-time filtering by discarding predicted-unsafe actions and retaining only those satisfying $\hat{c}(\mathbf{s},\mathbf{a}) = 0$. This provides a straightforward way to construct a safety filter; however, by itself, it does not guarantee safety. In this subsection, we introduce the filtering rule, and in the next subsection, we derive a sufficient condition for chance-constraint satisfaction.

We define a candidate action set $A_{\text{cand}}(\mathbf{s}) \subset A$ that captures most of the probability mass of the base policy $\pi_{\text{off}}(\cdot \mid \mathbf{s})$, i.e., $\int_{A_{\text{cand}}(\mathbf{s})} \pi_{\text{off}}(\mathbf{a} \mid \mathbf{s})\, d\mathbf{a} \ge 1 - \alpha$, for a small $\alpha \in (0,1)$. The predicted-safe action set is then given by

$$A_{\text{safe}}(\mathbf{s}) := \{\mathbf{a} \in A_{\text{cand}}(\mathbf{s}) \mid \hat{c}(\mathbf{s},\mathbf{a}) = 0\}.$$

We then construct a deployed policy π by renormalizing the base policy π_{off} over $A_{\text{safe}}(\mathbf{s})$ as

$$\pi(\mathbf{a} \mid \mathbf{s}) := \begin{cases} \dfrac{\pi_{\text{off}}(\mathbf{a} \mid \mathbf{s})}{\int_{A_{\text{safe}}(\mathbf{s})} \pi_{\text{off}}(\mathbf{a}' \mid \mathbf{s})\, d\mathbf{a}'} & \text{if } A_{\text{safe}}(\mathbf{s}) \neq \varnothing \text{ and } \mathbf{a} \in A_{\text{safe}}(\mathbf{s}), \\ 1 & \text{if } A_{\text{safe}}(\mathbf{s}) = \varnothing \text{ and } \mathbf{a} = \mathbf{a}_{\text{def}}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

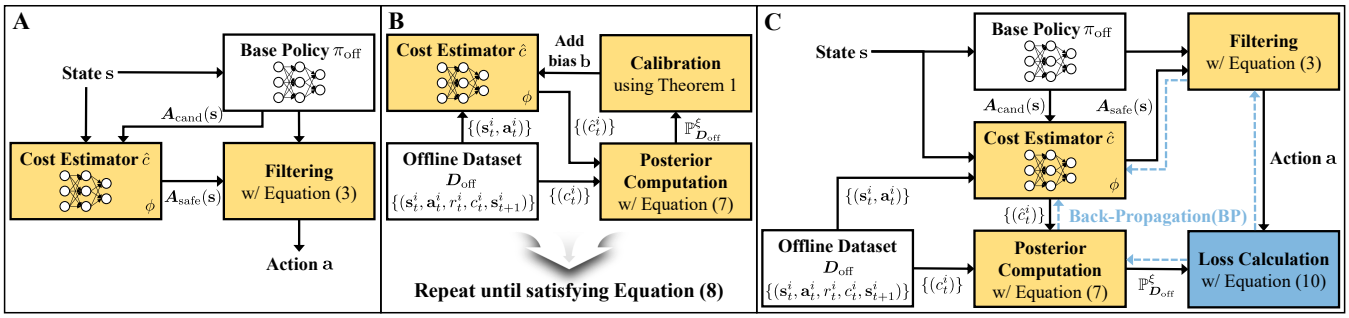

Fig. 1. Overview of our method. **A**. Deployment: Evaluating action candidates $A_{\text{cand}}(\mathbf{s})$ with the cost estimator to identify safe actions $A_{\text{safe}}(\mathbf{s})$, then renormalizing the base policy π_{off} over the safe set to select the final action. **B**. Calibration: Iteratively shifting the cost estimator's decision boundary by updating **b** based on posterior computation until the sufficient condition for the safety constraint is satisfied. **C**. Back-Propagation (BP): Fine-tuning the pretrained estimator via back-propagation to improve task performance.

If $A_{\text{safe}}(\mathbf{s}) = \varnothing$, we refer to $\mathbf{s}$ as an *infeasible state*. In such cases, we fall back to a predefined *default action* $\mathbf{a}_{\text{def}} \in A$ (e.g., sudden braking), which is designed to reduce immediate cost. This procedure is illustrated in Fig. 1-**A**.

Using a learned cost estimator to filter actions has been explored in prior safe RL work [24], [25], [26]. In these approaches, unsafe actions are rejected or penalized based on the predicted probability of constraint violation. However, such methods typically rely directly on the estimator's output and implicitly assume that the learned model is sufficiently accurate. Since the estimator is implemented as a neural network and trained from finite data, its prediction errors, particularly false negatives that classify unsafe actions as safe, can be non-negligible. Consequently, these approaches generally improve empirical safety but do not provide formal probabilistic guarantees at deployment.

C. Posterior Computation and Calibration

Our goal is to ensure that the deployed policy satisfies the chance constraint. However, the filtered policy π selects actions according to the predicted label $\hat{c}(\mathbf{s}, \mathbf{a})$, whereas the chance constraint (1) is defined with respect to the true cost $c(\mathbf{s}, \mathbf{a})$. These discrepancies may lead to unsafe decisions, since the cost estimator can overfit to the training data or encounter OOD inputs during real-world deployment. To mitigate this issue, we quantify the misclassification risk of the cost estimator under the deployed policy—namely, how often predicted-safe actions are actually unsafe—and use it to derive a sufficient condition for the chance constraint.

Let $\mathbb{P}_\pi[\cdot]$ and $\mathbb{P}_{D_{\text{off}}}[\cdot]$ denote probabilities computed with respect to the deployment state–action distribution d^π and the dataset state–action distribution $d^{D_{\text{off}}}$, respectively. For any measurable subset $E \subset S \times A$,

$$\mathbb{P}_\pi[E] := \int_E d^\pi(\mathbf{s}, \mathbf{a})\, d\mathbf{s}\, d\mathbf{a}, \quad \mathbb{P}_{D_{\text{off}}}[E] := \int_E d^{D_{\text{off}}}(\mathbf{s}, \mathbf{a})\, d\mathbf{s}\, d\mathbf{a}.$$

In particular, $\mathbb{P}_\pi[c = 1, \hat{c} = 0]$ denotes the probability of the set $\{(\mathbf{s}, \mathbf{a}) : c(\mathbf{s}, \mathbf{a}) = 1, \hat{c}(\mathbf{s}, \mathbf{a}) = 0\}$ under d^π, and similarly for $\mathbb{P}_{D_{\text{off}}}[c = 1, \hat{c} = 0]$. Also, we define the conditional deployment posterior as $\mathbb{P}_\pi[c = 1 \mid \hat{c} = 0] := \frac{\mathbb{P}_\pi[c=1, \hat{c}=0]}{\mathbb{P}_\pi[\hat{c}=0]}$, and analogously define $\mathbb{P}_{D_{\text{off}}}[c = 1 \mid \hat{c} = 0]$.

1) Relating $\mathbb{P}_\pi$ to $\mathbb{P}_{D_{off}}$: By the law of total probability, we decompose the violation probability as

$$\mathbb{P}_\pi[c = 1] = \underbrace{\mathbb{P}_\pi[c = 1, \hat{c} = 0]}_{\text{false negatives}} + \underbrace{\mathbb{P}_\pi[c = 1, \hat{c} = 1]}_{\text{default action term}}. \quad (4)$$

Under the proposed filtering rule (3), the policy π selects only actions with $\hat{c} = 0$ in all feasible states. Hence, the event $\hat{c} = 1$ arises solely from infeasible states where the default action $\mathbf{a}_{\text{def}}$ is executed. In particular, if $\hat{c}(\mathbf{s}, \mathbf{a}_{\text{def}}) = 0$ for all $\mathbf{s}$, then $\mathbb{P}_\pi[\hat{c} = 1] = 0$ and the second term vanishes. To accommodate infeasible states in which $\mathbf{a}_{\text{def}}$ may be predicted-unsafe, we upper-bound the second term by a small constant $\varepsilon \geq 0$.

Thus, to derive a sufficient condition for the chance constraint (1), it suffices to upper-bound $\mathbb{P}_\pi[c = 1, \hat{c} = 0]$. However, this quantity depends on the unknown deployment distribution d^π and is not directly evaluated offline. To overcome this difficulty, we relate it to the corresponding dataset probability $\mathbb{P}_{D_{\text{off}}}[c = 1, \hat{c} = 0]$, which can be estimated from D_{off}, via the following lemma.

Lemma 1. *For any measurable subset $E \subset S \times A$, let $W[E] := \frac{\mathbb{P}_\pi[E]}{\mathbb{P}_{D_{off}}[E]}$. Then, for $\overline{W} \geq W[c = 1, \hat{c} = 0]$, we have*

$$\mathbb{P}_\pi[c = 1, \hat{c} = 0] \leq \overline{W} \cdot \mathbb{P}_{D_{off}}[c = 1, \hat{c} = 0]. \quad (5)$$

Proof: It follows from the definitions of W and $\overline{W}$. □

Different from the stationary distribution correction term $w(s, a) = \frac{d^\pi(s,a)}{d^{D_{\text{off}}}(s,a)}$, used in distribution-correction methods [6], which is defined pointwise for each state–action pair, our quantity $W[E]$ is an event-level probability ratio defined over a measurable set E.

By Lemma 1, it suffices to upper-bound the dataset-probability term (the right-hand side of (5)) to derive a sufficient condition for the chance constraint. In particular, Lemma 1 with (4) yields

$$\mathbb{P}_\pi[c = 1] \leq \overline{W} \cdot \mathbb{P}_{D_{\text{off}}}[\hat{c} = 0] \cdot \mathbb{P}_{D_{\text{off}}}[c = 1 \mid \hat{c} = 0] + \varepsilon. \quad (6)$$

However, directly using empirical estimates from the offline dataset can be unreliable for safety-critical decisions because the offline dataset is finite and may be biased. Among the two dataset-dependent factors, $\mathbb{P}_{D_{\text{off}}}[\hat{c} = 0]$ and $\mathbb{P}_{D_{\text{off}}}[c = 1 \mid$

$\hat{c}=0]$, we note that the latter, which corresponds to the false-negative rate of the cost estimator, directly amplifies safety risk and thus governs the probability of safety violation.

2) Computing Conservative Posteriors: To conservatively estimate the dataset-level violation probability, we adopt the conservative posterior computation of [12]. Specifically, we use its conservative counting mechanism, which perturbs the model outputs within a prescribed neighborhood and computes worst-case conditional probabilities, to construct an upper bound for the posterior $\mathbb{P}_{D_{\text{off}}}[c=1 \mid \hat{c}=0]$. We then use this bound to derive a sufficient condition tailored to our offline RL policy-filtering problem.

For each cost label $k \in C$, we define

$$I_{D_{\text{off}}}(c=k) := \{(i,t) \mid c_t^i = k\}, \quad N_{D_{\text{off}}}(c=k) := |I_{D_{\text{off}}}(c=k)|.$$

Following [12], we define the following count functions:

$$N^{+\xi}_{D_{\text{off}}}(c=k,\hat{c}=j) := \sum_{(i,t)\in I_{D_{\text{off}}}(c=k)} \max_{\mathbf{z}\in B(f_\phi(\mathbf{s}_t^i,\mathbf{a}_t^i))} \mathbb{I}(g(\mathbf{z})=j),$$

$$N^{-\xi}_{D_{\text{off}}}(c=k,\hat{c}=j) := \sum_{(i,t)\in I_{D_{\text{off}}}(c=k)} \min_{\mathbf{z}\in B(f_\phi(\mathbf{s}_t^i,\mathbf{a}_t^i))} \mathbb{I}(g(\mathbf{z})=j),$$

where $B(f_\phi(\mathbf{s}_t^i,\mathbf{a}_t^i))$ is the closed ξ-ball around $f_\phi(\mathbf{s}_t^i,\mathbf{a}_t^i)$ in the model output space $\mathbb{R}^{n_c}$, and $j \in C$ denotes the predicted cost label. Intuitively, the parameter ξ controls how conservatively samples are counted by allowing the predicted label to vary within a ξ-ball in logit space. Applying Theorem 1 of [12], the posterior $\mathbb{P}_{D_{\text{off}}}[c=k \mid \hat{c}=j]$ admits the following conservative upper bound, where the numerator is maximized and the denominator is minimized within the ξ-ball:

$$\mathbb{P}^{\xi}_{D_{\text{off}}}[c=k \mid \hat{c}=j] := \frac{\mathbb{P}^{+\xi}_{D_{\text{off}}}[\hat{c}=j \mid c=k]\cdot \text{prior}(c=k)}{\sum_{q=0}^{1}\mathbb{P}^{-\xi}_{D_{\text{off}}}[\hat{c}=j \mid c=q]\cdot \text{prior}(c=q)}, \tag{7}$$

where $\mathbb{P}^{\pm\xi}_{D_{\text{off}}}[\hat{c}=j \mid c=q] := \frac{N^{\pm\xi}_{D_{\text{off}}}(c=q,\hat{c}=j)}{N_{D_{\text{off}}}(c=q)}$. Here, $\text{prior}(c=q)$ denotes the prior probability for $c(\mathbf{s},\mathbf{a})=q$. The prior values are summarized in Section I of the supplementary material.[1] Based on this conservative posterior bound, we derive the following sufficient condition for the chance constraint (1).

Theorem 1. *For the given $\varepsilon \geq 0$, if*

$$\mathbb{P}^{\xi}_{D_{off}}[c=1 \mid \hat{c}=0] \leq \delta' := \frac{\delta-\varepsilon}{\overline{W}\cdot \mathbb{P}_{D_{off}}[\hat{c}=0]} \tag{8}$$

then constraint (1) *is satisfied.*

Proof: By construction of the conservative bound in (7)*, we have $\mathbb{P}_{D_{off}}[c=1 \mid \hat{c}=0] \leq \mathbb{P}^{\xi}_{D_{off}}[c=1 \mid \hat{c}=0]$. Then, we can rewrite the inequality* (6) *if* (8) *holds as*

$$\begin{aligned}\mathbb{P}_\pi[c=1] &\leq \overline{W}\cdot \mathbb{P}_{D_{off}}[\hat{c}=0]\cdot \frac{\delta-\varepsilon}{\overline{W}\cdot \mathbb{P}_{D_{off}}[\hat{c}=0]} + \varepsilon \\ &= \delta-\varepsilon+\varepsilon = \delta. \end{aligned}$$ □

In implementation, we treat $\overline{W}$ and ε as environment-dependent parameters, which eliminates the explicit dependence on π in (8). In particular, we select $\overline{W}$ using a task-specific upper bound based on empirical observations, as detailed in Section VI. We estimate $\mathbb{P}_{D_{\text{off}}}[\hat{c}=0]$ using empirical frequencies in D_{off}. Through this procedure, while π_{off} is learned from D_{off}, our approach further reuses the same dataset to estimate a conservative posterior bound used to construct the deployed policy π.

3) Calibration: To enforce the sufficient condition (8), we perform an offline, post-training calibration of the pretrained cost estimator using the bias-correction method [12]. Specifically, we add a constant bias vector $\mathbf{b} \in \mathbb{R}^{n_c}$ to the final-layer logits: $\hat{c}(\mathbf{s},\mathbf{a}) = g\big(f_\phi(\mathbf{s},\mathbf{a})+\mathbf{b}\big)$. This bias shifts the decision boundary of the estimator, thereby controlling how conservatively actions are labeled as safe. This procedure is illustrated in Fig. 1-**B**. For a target safety threshold δ, we choose $\mathbf{b}$ such that (8) holds, which ensures that the resulting filtered policy satisfies the chance constraint $\mathbb{P}_\pi[c=1] \leq \delta$. Adapting to a new threshold δ at deployment only requires recomputing $\mathbf{b}$ without retraining the cost estimator. Thus, even with an imperfect cost estimator, offline calibration supports constraint satisfaction while preserving zero-shot adaptability to varying safety thresholds.

D. Improvement via Back-propagation

While calibrating the cost estimator guarantees satisfaction of the chance constraint, it can be overly conservative and may degrade performance. In particular, the bias-correction step shifts the decision boundary solely to enforce (8) without explicitly accounting for reward optimization.

To alleviate this conservativeness, we further fine-tune the cost estimator via back-propagation while keeping the base policy π_{off} fixed. This procedure is illustrated in Fig. 1-**C**. Let $J(\mathbf{a};\mathbf{s})$ denote our objective function, which guides the optimization toward actions preferred by the offline policy π_{off}. To fine-tune the cost estimator, we adapt the approximated general loss construction of [12] to our policy-filtering objective:

$$\tilde{L}(j) = \min_{\mathbf{a}\in A}\Big(J(\mathbf{a};\mathbf{s}) - \beta \min\big(\delta' - \mathbb{P}^{\xi}_{D_{\text{off}}}[c=1 \mid \hat{c}=j], 0\big)\Big). \tag{9}$$

This objective is a penalty-based loss that remains well-behaved with respect to the estimator's continuous outputs (e.g., logits or scores). It can be viewed as a relaxation of the original constrained problem, in which $J(\mathbf{a};\mathbf{s})$ is minimized subject to a constraint on the posterior probability. Proposition 5 of [12] shows that once β exceeds a certain threshold, any solution to (9) lies within a prescribed distance of a solution to the original constrained problem.

Note that $\tilde{L}$ still contains a non-smooth component, since the constraint term in (7) depends on the discrete estimator output $\hat{c}(\mathbf{s},\mathbf{a}) = j \in \{0,1\}$. To address this issue, we define a softened loss by taking the expectation of $\tilde{L}$ with respect to the softmax output p, yielding the final training loss

$$L = \sum_{j\in\{0,1\}} p(j; f_\phi(\mathbf{s},\mathbf{a}))\cdot \tilde{L}(j). \tag{10}$$

Moreover, to ensure differentiability of f_ϕ, we replace the max operator in (2) with a log-sum-exp surrogate. Details of

[1]The supplementary material is available on the project page at `https://safeofflinerl-cpf.github.io/`.

the back-propagation, including the specific definition of J, are summarized in Section II of the supplementary material.

V. Applications

Our method serves as an inference-time safety filter and can be readily integrated with existing offline RL algorithms. In particular, we apply our framework as a deployment-time module on top of BCQ [13] and BEAR [14]. Although we have so far treated the action space as continuous, exhaustively filtering over $A_{\text{cand}}(\mathbf{s})$ is impractical. Instead, at each state $\mathbf{s}$, we apply the filter to a finite set of candidate actions that sufficiently covers the probability mass of the base policy $\pi_{\text{off}}(\cdot \mid \mathbf{s})$. The detailed evaluation procedures are provided in Section III of the supplementary material.

A. Candidate Actions and Filtering

For each state $\mathbf{s}$, we construct a discrete candidate action set $\bar{A}_{\text{cand}}(\mathbf{s}) = \{\bar{\mathbf{a}}^{(1)}, \ldots, \bar{\mathbf{a}}^{(n)}\} \subset A_{\text{cand}}(\mathbf{s})$. We then forward each candidate action to the (calibrated) cost estimator and define the discrete predicted-safe action set as $\bar{A}_{\text{safe}}(\mathbf{s}) := \{\bar{\mathbf{a}} \in \bar{A}_{\text{cand}}(\mathbf{s}) \mid \hat{c}(\mathbf{s}, \bar{\mathbf{a}}) = 0\}$. When $\bar{A}_{\text{safe}}(\mathbf{s}) \neq \varnothing$, we deploy a filtered policy by renormalizing the base policy over the predicted-safe candidates:

$$\pi(\bar{\mathbf{a}} \mid \mathbf{s}) = \frac{\pi_{\text{off}}(\bar{\mathbf{a}} \mid \mathbf{s})}{\sum_{\bar{\mathbf{a}}' \in \bar{A}_{\text{safe}}(\mathbf{s})} \pi_{\text{off}}(\bar{\mathbf{a}}' \mid \mathbf{s})} \cdot \mathbb{I}\{\bar{\mathbf{a}} \in \bar{A}_{\text{safe}}(\mathbf{s})\}.$$

B. Instantiation for BEAR and BCQ

In BEAR, the actor outputs the mean $\mu(\mathbf{s}) = [\mu_x(\mathbf{s}), \mu_y(\mathbf{s})]$ and the standard deviation of a Gaussian distribution, and samples actions from this distribution. Thus, when we use BEAR as π_{off}, we form $\bar{A}_{\text{cand}}(\mathbf{s})$ by uniformly discretizing the box $[\mu_x(\mathbf{s}) - 2,\ \mu_x(\mathbf{s}) + 2] \times [\mu_y(\mathbf{s}) - 2,\ \mu_y(\mathbf{s}) + 2]$ with grid interval $\Delta = 0.04$ for each axis, so that the candidates cover the high-probability region of $\pi_{\text{off}}(\cdot \mid \mathbf{s})$. We then define $\pi_{\text{off}}(\bar{\mathbf{a}} \mid \mathbf{s})$ on this grid by assigning each $\bar{\mathbf{a}} \in \bar{A}_{\text{cand}}(\mathbf{s})$ the probability mass of its corresponding grid cell using the mean and standard deviation output by the actor, yielding a histogram approximation of $\pi_{\text{off}}(\cdot \mid \mathbf{s})$.

In contrast, BCQ generates actions by sampling a latent vector $\mathbf{z}$ and decoding it through a variational autoencoder (VAE), followed by perturbation. Thus, instead of discretizing actions, we discretize the latent space, decode it into actions via the VAE, and apply perturbation to obtain $\bar{A}_{\text{cand}}(\mathbf{s})$.

VI. Experiments

A. Experimental Setup

1) Environment (Tasks and Robots): We evaluate our method on the `Goal` tasks using the `Car` and `Point` robots in the OSRL benchmark [27]. In each `Goal` task (`CarGoal1`, `CarGoal2`, `PointGoal1`, `PointGoal2`), the agent must reach a sequence of goal positions while avoiding hazards. The only difference between `Goal1` and `Goal2` is the number of cost-incurring objects (hazards and vases), with `Goal2` containing more. The `Car` receives a 72-dimensional observation vector and executes a two-dimensional continuous action in the range $[-1, 1]$, corresponding to the forces applied to the left and right wheels, respectively. The `Point` robot receives a 60-dimensional observation vector and executes a two-dimensional action in the range $[-1, 1]$, corresponding to the force applied to translational and rotational motions, respectively.

2) Baselines and Base Policies: We compare our method against BCQ [13]+Lagrangian (BCQL), BEAR [14]+Lagrangian (BEARL), CDT [7], and CAPS [24], all implemented using the default OSRL settings. For CDT, we use a sequence length of 40. Each model is trained up to a task-specific budget $T_{\text{total}} \in \{40, 60, 80\} \times 10^4$ epochs, with checkpoints evaluated every 10^4 epochs. For Table I, baseline results are obtained by selecting the checkpoint with the highest reward for each seed in $\{0, 10, 20\}$, and then averaging the corresponding returns across seeds. Details on the selection of baselines are summarized in Section IV of the supplementary material.

For the base policy π_{off}, we train BCQ and BEAR under the same budget. We select the base policy checkpoint (at epoch T_{base}) based on the highest mean reward averaged over the three seeds.

3) Cost Estimator:

a) Pretraining: After selecting a base-policy checkpoint at epoch T_{base} (shared across seeds), we pretrain a separate cost estimator for each seed using D_{off} for the remaining budget $T_{\text{total}} - T_{\text{base}}$ epochs, ensuring that the total training budget matches that of CAPS and CDT. For each seed, we select the cost estimator checkpoint that achieves the highest average reward over 20 evaluation episodes with the fixed base policy. In Table I, *Ours(PT)* reports the mean reward and cost across the three seeds.

b) Back-propagation: Starting from the selected cost estimator checkpoint in Ours(PT) for each seed, we fine-tune the estimator for 100×10^3 epochs, saving checkpoints every 2×10^3 epochs. For each seed, we select the checkpoint that achieves the highest average reward return among those evaluated during fine-tuning. *Ours(BP)* in Table I reports the mean reward and cost returns across the three seeds.

Details of the network architecture, hyperparameters, and the selected checkpoints are provided in Section V of the supplementary material.

4) Parameter Setting ($\varepsilon, \overline{W}$): For the `Car`, the default action rarely leads to hazard entry in infeasible states; therefore, we set $\varepsilon = 0$. In contrast, for the `Point`, we set $\varepsilon = 0.018$, as sudden braking is less effective.

We set $\overline{W}$ in a task-dependent manner for `Goal1` and `Goal2`. We approximate $W[c = 1, \hat{c} = 0] \approx \frac{\mathbb{P}_\pi[c=1]}{\mathbb{P}_{D_{\text{off}}}[c=1]}$, which amounts to assuming that the false-negative rate $\mathbb{P}[\hat{c} = 0 \mid c = 1]$ does not vary drastically between deployment and the offline dataset. Here, we estimate $\mathbb{P}_{D_{\text{off}}}[c = 1]$ using empirical frequencies in D_{off}. For $\mathbb{P}_\pi[c = 1]$, which is the target quantity in the chance constraint and is unknown at deployment time, we use a heuristic, task-specific upper bound based on empirical observations of the environment. Empirically, in `Goal1`, the number of steps spent in hazardous regions rarely exceeds 100 out of 1000 per episode, whereas in `Goal2` it rarely exceeds 150. Accordingly, we set $\overline{W} =$

TABLE I

EVALUATION RESULTS OF THE PROPOSED METHOD AND BASELINES

Task	$1000 \cdot \delta$	CDT		CAPS(SAC)		BCQL		BEARL		BCQ+Ours(PT)		BEAR+Ours(PT)		BEAR+Ours(BP)	
		R	C	R	C	R	C	R	C	R	C	R	C	R	C
CarGoal1	20	27.35	36.60	19.58	21.28	22.31	30.63	29.05	41.58	**11.97**$_{\pm 2.14}$	**19.20**$_{\pm 9.89}$	**11.09**$_{\pm 2.30}$	**16.07**$_{\pm 8.82}$	**12.45**$_{\pm 1.91}$	**17.93**$_{\pm 9.81}$
	40	28.00	40.37	**19.59**	**21.28**	**23.01**	**29.03**	28.64	48.13	**14.02**$_{\pm 2.11}$	**15.45**$_{\pm 3.65}$	**14.35**$_{\pm 2.12}$	**27.58**$_{\pm 1.50}$	**17.34**$_{\pm 1.37}$	**31.90**$_{\pm 4.38}$
	80	**28.44**	**42.18**	**19.49**	**24.42**	**22.34**	**20.48**	**28.80**	**41.60**	**20.49**$^{*}_{\pm 1.03}$	**24.92**$^{*}_{\pm 7.75}$	**22.42**$^{*}_{\pm 2.38}$	**34.82**$^{*}_{\pm 7.88}$	-	-
CarGoal2	20	15.64	71.63	9.97	40.95	12.03	69.63	21.36	123.00	**6.85**$_{\pm 0.89}$	**14.22**$_{\pm 3.05}$	**6.01**$_{\pm 0.49}$	**9.98**$_{\pm 2.83}$	**5.13**$_{\pm 0.72}$	**8.10**$_{\pm 0.44}$
	40	16.67	67.18	9.96	40.95	12.32	51.50	21.78	120.55	**6.60**$_{\pm 0.07}$	**21.93**$_{\pm 16.41}$	**7.25**$_{\pm 1.24}$	**14.27**$_{\pm 6.00}$	**8.57**$_{\pm 1.25}$	**22.33**$_{\pm 4.77}$
	80	18.41	87.33	**9.85**	**47.92**	**11.80**	**56.27**	21.05	133.78	**8.05**$_{\pm 0.43}$	**23.55**$_{\pm 2.41}$	**7.90**$_{\pm 0.71}$	**20.05**$_{\pm 5.90}$	**12.27**$_{\pm 0.40}$	**54.28**$_{\pm 9.45}$
PointGoal1	20	21.82	28.85	**9.29**	**10.73**	22.25	43.37	23.62	35.52	**3.49**$_{\pm 0.38}$	**3.13**$_{\pm 0.87}$	**4.92**$_{\pm 0.81}$	**2.55**$_{\pm 0.95}$	**3.43**$_{\pm 0.43}$	**0.03**$_{\pm 0.06}$
	40	22.39	43.78	**13.36**	**23.12**	**22.50**	**37.17**	23.34	43.00	**21.70**$^{*}_{\pm 0.37}$	**39.08**$^{*}_{\pm 8.99}$	**17.52**$^{*}_{\pm 0.46}$	**29.92**$^{*}_{\pm 3.84}$	-	-
	80	**22.46**	**40.73**	**14.91**	**31.60**	**22.31**	**41.07**	**23.33**	**49.47**	**21.78**$^{*}_{\pm 0.32}$	**44.68**$^{*}_{\pm 3.21}$	**18.95**$^{*}_{\pm 0.23}$	**37.45**$^{*}_{\pm 5.47}$	-	-
PointGoal2	20	16.97	90.22	9.72	44.13	21.12	114.57	22.80	123.03	**0.48**$_{\pm 0.11}$	**0.00**$_{\pm 0.00}$	**0.55**$_{\pm 0.42}$	**0.00**$_{\pm 0.00}$	**0.46**$_{\pm 0.31}$	**0.00**$_{\pm 0.00}$
	40	17.67	92.47	**9.21**	**36.55**	20.92	128.75	22.74	122.70	**7.35**$_{\pm 0.38}$	**35.27**$_{\pm 3.26}$	**6.60**$_{\pm 1.19}$	**24.35**$_{\pm 8.79}$	**6.22**$_{\pm 0.89}$	**25.95**$_{\pm 13.36}$
	80	18.73	95.50	**11.56**	**59.28**	20.83	107.07	22.58	130.67	**9.76**$_{\pm 0.74}$	**49.87**$_{\pm 10.57}$	**9.14**$_{\pm 0.37}$	**48.93**$_{\pm 7.79}$	**12.06**$_{\pm 0.67}$	**61.82**$_{\pm 20.84}$

1) R denotes the average reward return and C denotes the average cost return. Values with the $\pm$ subscript are reported as mean$_{\pm \text{std}}$ over three seeds.
2) **Bold** numbers denote results satisfying the chance constraint, while **blue** numbers mark the highest reward among them under the same threshold.
3) '*' indicates cases where approximately 98% of action candidates are classified as safe, yielding behavior nearly identical to the base policy. Since the filter induces little to no change in action selection, back-propagation is not meaningful and is not performed; this is denoted by '-'.

$\frac{0.100}{\mathbb{P}_{D_{\text{off}}}[c=1]}$ for `Goal1` and $\overline{W} = \frac{0.150}{\mathbb{P}_{D_{\text{off}}}[c=1]}$ for `Goal2`. Larger $\overline{W}$ corresponds to a more conservative assumption about distribution shift (i.e., potentially higher deployment risk than in the offline dataset), which tightens the safety constraint.

B. Results

As shown in Table I, Ours(PT) satisfies varying safety thresholds ($\delta \in \{0.02, 0.04, 0.08\}$) during deployment without additional training across multiple offline base policies (BCQ and BEAR). The constraint is satisfied in all settings (marked in **bold**), whereas the strongest reported baseline, CAPS, satisfies it in 8 out of 12 settings. These results demonstrate that our approach successfully satisfies the prescribed chance constraint across all four tasks. CDT and CAPS can also handle different safety thresholds at deployment time without additional training, whereas the Lagrangian baselines (BCQL and BEARL) require retraining for each threshold. Accordingly, we separate the BCQL/BEARL results with horizontal lines to indicate independently retrained runs for each threshold.

To better understand the behavior of the filter, we examine `CarGoal1` and `PointGoal1`. In 6 out of the 12 reported Ours(PT) results, over 98% of action candidates are classified as safe (marked with '*'). In these cases, the safety filter induces negligible change, and the performance remains nearly identical to that of the base policy.

In Ours(BP), back-propagation through the cost estimator yields modest performance gains while preserving safety. These improvements are more pronounced in `CarGoal2` and `PointGoal2`, which contain no '*' cases and therefore offer greater room for improvement. Overall, this tuning maintains safe behavior across all settings and achieves the highest reward return in more tasks than the baselines (i.e., more **blue** entries). These findings indicate that our method empirically satisfies the prescribed constraints more consistently across varying safety thresholds.

C. Deployment with Inaccurate Cost Estimator

We evaluate whether our method preserves constraint satisfaction under the target threshold $\delta = 0.02$ even when the estimator is imperfect, i.e., has low recall. We pretrain the cost estimator for 30×10^3 epochs and save checkpoints every 2×10^3 epochs, yielding 15 estimator checkpoints per task. For each estimator checkpoint, Fig. 2 reports the reward and cost returns averaged over three seeds, using 20 episodes per seed (circles/squares). For reference, we also mark the best checkpoints used in the main experiments (star/plus).

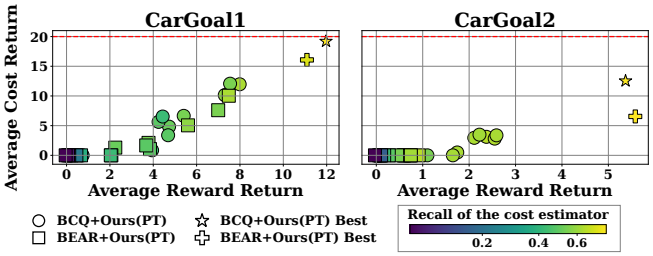

Fig. 2. Deployment under inaccurate cost estimators. Circles and squares indicate deployment performance using cost estimator checkpoints obtained at earlier training stages, while the star and plus denote the best checkpoints.

All evaluated estimator checkpoints satisfy the safety constraint. When the cost estimator has low recall, the deployed policies operate well below the safety threshold. This behavior reflects increased conservatism, as our filtering rule enforces an explicit upper bound on the misclassification risk. The best checkpoints (star/plus), which correspond to higher recall, require a smaller conservatism margin and therefore achieve higher reward while satisfying the safety constraint. Overall, these results indicate that improved cost estimation induces less conservative decisions, leading to higher reward return while still satisfying the safety constraint. Further experimental details are provided in Section VI of the supplementary material.

D. Sensitivity Analysis of ε and $\overline{W}$

As described in Section VI-A.4, we set ε and $\overline{W}$ using heuristic estimates based on empirical observations of the environment and the offline dataset. These quantities are meant to capture the residual default-action risk and the distribution shift, respectively. Thus, in practice, the safety

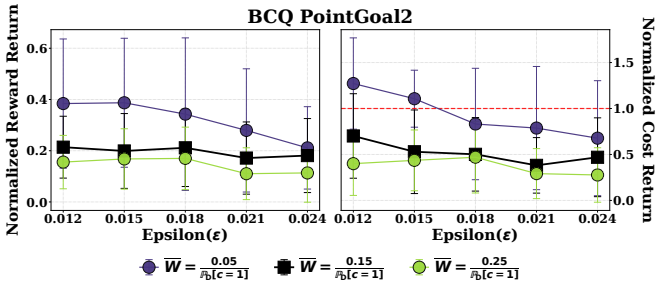

Fig. 3. Sensitivity analysis of ε and $\overline{W}$ on `PointGoal2` using BCQ. We report normalized reward and cost returns following the OSRL benchmark convention [27], where reward is normalized by task-level empirical reward ranges and cost is normalized by the target threshold multiplied by the episode length, $1000 \cdot \delta \in \{20, 40, 80\}$. Each point shows the mean over nine values from three seeds and three target safety thresholds, with error bars indicating the corresponding standard deviation.

guarantee of the proposed method is conditional on the validity of these assumed or estimated bounds.

To assess the effect of these bounds, we present a sensitivity analysis on `PointGoal2`, since this task is more sensitive to both ε and $\overline{W}$ than the other tasks. In Fig. 3, we vary ε along the x-axis and plot separate curves for different values of $\overline{W}$. The default setting used in the main experiments corresponds to $\varepsilon = 0.018$ and the black curve, $\overline{W} = \frac{0.150}{\mathbb{P}_{D_{\text{off}}}[c=1]}$.

The results show that overly small values of ε or $\overline{W}$, which correspond to underestimating the residual default-action risk or the distribution shift, can cause constraint violation. For instance, the purple curve with $\overline{W} = \frac{0.05}{\mathbb{P}_{D_{\text{off}}}[c=1]}$ frequently exhibits constraint violation when $\varepsilon = 0.012$ or 0.015. In contrast, more conservative values of ε and $\overline{W}$ keep the cost return below the constraint threshold, at the cost of reduced reward return.

VII. Conclusion

This paper presented a chance-constrained policy filtering method for safe offline RL that aims to satisfy user-specified probabilistic safety constraints at deployment. The method employs a learned cost estimator to construct a safety-filtered policy that selectively accepts or rejects candidate actions, while remaining modular and readily compatible with existing offline RL pipelines. Unlike prior approaches that directly rely on learned estimators, our method does not assume that the cost estimator is perfectly optimized. Instead, we explicitly account for possible estimation errors and constrain it by calibrating the estimator via posterior computation and bias-correction. To improve reliability under finite and potentially biased data, conservative testing is employed to obtain a conservative bound on the true safety-violation probability. The framework can also be incorporated into end-to-end back-propagation, yielding additional performance gains while preserving safety. Experimental results on the OSRL benchmark show that the proposed framework can satisfy chance constraints using only offline data, supports deployment under varying safety thresholds without retraining, and maintains safety under the assumed bounds even when the cost estimator is imperfect.

References

[1] D. Mayne, M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, 2005.

[2] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, 2024.

[3] S. E. Li, *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer, 2023.

[4] S. Levine, *et al.*, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, arXiv:2005.01643.

[5] H. Xu, X. Zhan, and X. Zhu, "Constraints penalized q-learning for safe offline reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[6] J. Lee, *et al.*, "Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation," in *International Conference on Learning Representations*, 2022.

[7] Z. Liu, *et al.*, "Constrained decision transformer for offline safe reinforcement learning," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[8] Y. Yao, *et al.*, "Oasis: Conditional distribution shaping for offline safe reinforcement learning," in *Advances in Neural Information Processing Systems*, 2024.

[9] Y. Zheng, *et al.*, "Safe offline reinforcement learning with feasibility-guided diffusion model," in *International Conference on Learning Representations*, 2024.

[10] H. Zhou, Y. Zhang, and W. Luo, "Computationally and sample efficient safe reinforcement learning using adaptive conformal prediction," in *IEEE International Conference on Robotics and Automation*, 2025.

[11] A. C. Stutts, *et al.*, "Uncertainty-aware deep reinforcement learning with calibrated quantile regression and evidential learning," in *IEEE International Conference on Robotics and Automation*, 2025.

[12] B. Kim, *et al.*, "A domain-agnostic scalable ai safety ensuring framework," 2025, arXiv:2504.20924.

[13] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[14] A. Kumar, *et al.*, "Stabilizing off-policy q-learning via bootstrapping error reduction," in *Advances in Neural Information Processing Systems*, 2019.

[15] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," 2019, arXiv:1911.11361.

[16] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Advances in Neural Information Processing Systems*, 2021.

[17] R. Zhang, *et al.*, "Gendice: Generalized offline estimation of stationary values," in *International Conference on Learning Representations*, 2020.

[18] O. Nachum, *et al.*, "Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections," in *Advances in Neural Information Processing Systems*, 2019.

[19] J. Lee, *et al.*, "Optidice: Offline policy optimization via stationary distribution correction estimation," in *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[20] L. Mao, *et al.*, "Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, 2024.

[21] A. Kumar, *et al.*, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, 2020.

[22] L. Chen, *et al.*, "Decision transformer: Reinforcement learning via sequence modeling," in *Advances in Neural Information Processing Systems*, 2021.

[23] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems*, 2021.

[24] Y. Chemingui, *et al.*, "Constraint-adaptive policy switching for offline safe reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[25] H. Bharadhwaj, *et al.*, "Conservative safety critics for exploration," in *International Conference on Learning Representations*, 2021.

[26] K. Srinivasan, *et al.*, "Learning to be safe: Deep rl with a safety critic," 2020, arXiv:2010.14603.

[27] Z. Liu, *et al.*, "Datasets and benchmarks for offline safe reinforcement learning," *Journal of Data-centric Machine Learning Research*, 2024.