Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079959

ARTICLE

Safe Robot Control through Multi-Task Offline Reinforcement Learning with Multi-Scale Distribution Debiasing

Chengjing Li[1], **Li Wang**[2,*] **and Xiaoyan Zhao**[2]

[1]College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology, Jinzhong, China
[2]College of Artificial Intelligence, Taiyuan University of Technology, Jinzhong, China

*Corresponding Author: Li Wang. Email: wangli@tyut.edu.cn

Received: 31 January 2026; Accepted: 24 March 2026; Published: 08 May 2026

ABSTRACT: Robots perform diverse tasks in real-world scenarios. In safety-critical applications, robot control must prioritize satisfying safety constraints in addition to achieving high performance. Offline safe reinforcement learning avoids risky online exploration by training from a given dataset. However, most existing methods overlook two issues in offline data. First, non-zero cost signals are typically sparse, which leads to inaccurate cost value estimates and makes it difficult to impose effective safety constraints on the policy. Second, an imbalanced dataset biases policy learning toward unsafe behaviors. To address these challenges, we propose an actor-critic method ARMOR (multi-sc**A**le **R**eweighting with **M**ulti-task **O**ffline c**R**itic). The multi-task critic treats reward, long-term cost, and short-term cost as multiple tasks, learns shared representations to capture common state information, and leverages dense reward signals to stabilize learning under sparse cost signals. To mitigate dataset imbalance, ARMOR performs counterfactual reasoning with the short-term cost to upweight critical safe transitions near the risk boundary and assigns higher weights to low-cost trajectories. It then performs multi-scale reweighting by combining transition-level and trajectory-level weights to debias data distribution and emphasize safe demonstrations. The actor is parameterized by a conditional diffusion policy and trained via weighted behavior cloning. ARMOR additionally incorporates a reward-guided objective and a long-term cost constraint to improve the reward-cost trade-off. Extensive experiments on continuous-control robot tasks show that ARMOR achieves competitive performance under safety constraints, with clear advantages in several challenging environments. Furthermore, ARMOR exhibits zero-shot adaptation capability, making it suitable for practical deployment.

KEYWORDS: Robotics; offline safe reinforcement learning; diffusion model; imbalanced data

1 Introduction

Robot control is a key technology for autonomous systems. It is increasingly deployed in safety-critical domains, including robotic surgery [1], industrial automation [2], and Automated Guided Vehicle (AGV) transportation [3], where failures may cause serious harm. Therefore, ensuring safety satisfaction during decision-making is a priority requirement for real-world deployment. Reinforcement learning (RL) has received a great deal of attention in the field of robotic control [4], but online exploration can be unsafe and expensive. Offline safe reinforcement learning (OSRL) addresses this concern by learning from fixed datasets without risky interaction.

In OSRL, the offline dataset is collected by one or more behavior policies and consists of a set of trajectories. Each trajectory is a sequence of transitions, each of which provides a reward signal and a cost

signal. The reward quantifies task performance, while the cost represents safety violations, such as collisions, damage, or entering hazardous regions. The objective of OSRL is to learn a policy that maximizes the expected reward while satisfying a predefined cost limit.

Offline safe reinforcement learning faces two major challenges. Firstly, at the transition level, non-zero cost signals are typically sparse because safety violations do not occur at every time step, as shown in Fig. 1a. Consequently, cost value function learning is dominated by zero-cost samples, which leads to the underestimation of cost estimates and further makes it difficult to enforce constraints during policy optimization. Secondly, at the trajectory level, offline datasets are often imbalanced. Trajectories that satisfy the cost limit are rare, which hinders the learning of safe policies, as shown in Fig. 1b. To quantify these two challenges, we report dataset statistics for the public offline safe RL datasets released in [5]. The datasets are collected by a suite of policies, which are trained with different cost constraints using various safe RL algorithms. Table 1 reports the number of transitions, the non-zero cost transition rate P_{nz}, and the feasible trajectory ratio P_{feas} for each task, where feasible trajectories refer to trajectories whose cumulative cost satisfies the safety constraint. These statistics directly confirm that sparse cost supervision and dataset imbalance are pervasive in offline safe RL.

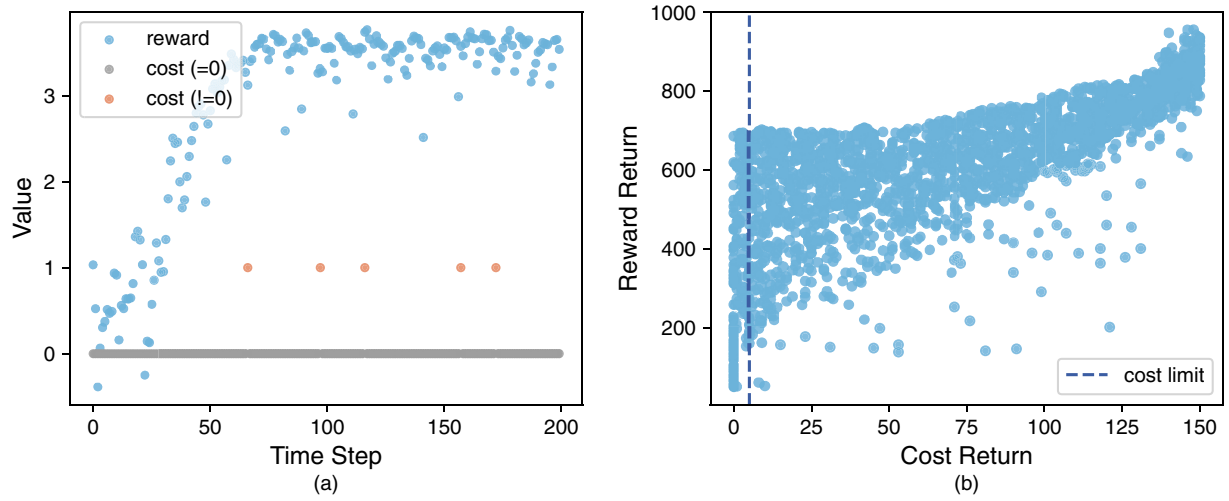

Figure 1: Visualization of reward and cost for the `AntRun` task, based on the datasets in [5]. (**a**) Each point represents (step, cost) or (step, reward) of a transition in an episode. (**b**) Each point represents (cost return, reward return) of a trajectory in the dataset. Only data on the left side of the dashed line is feasible.

Table 1: Offline dataset statistics across tasks.

Tasks	Transitions	P_{nz}	P_{feas}	Tasks	Transitions	P_{nz}	P_{feas}
Ant-Run	363,010	34.05%	8.04%	PointGoal1	2,022,000	3.07%	45.25%
Ball-Run	94,000	48.68%	4.47%	PointPush1	2,37,9000	5.31%	28.12%
Drone-Run	395,099	25.44%	23.42%	PointButton1	2,268,000	7.67%	20.28%
Ant-Velocity	2,092,264	11.74%	8.18%	CarCircle1	635,500	26.89%	7.39%
Walker2d-Velocity	2,121,658	12.33%	21.73%	CarGoal1	1,671,000	4.32%	32.79%
HalfCheetah-Velocity	2,495,000	11.54%	8.86%	CarPush1	2,871,000	5.55%	37.65%
PointCircle1	549,000	22.62%	12.57%	CarButton1	2,656,000	10.85%	13.06%

Accurate value estimation is a central challenge in offline reinforcement learning. Prior offline RL methods mitigate value estimation errors through behavior-regularized policy learning that restricts actions to the given data and conservative value estimation [6,7]. Building on these ideas, offline safe RL further introduces a cost critic and optimizes a Lagrangian objective. Primal-Dual-Critic Algorithm (PDCA) runs a primal-dual procedure over a critics-estimated Lagrangian [8]. Constrained Offline Policy Optimization (COPO) applies an offline cost-projection with confidence bounds to better account for distributional shift [9]. Other approaches explicitly handle out-of-distribution (OOD) behaviors for safety. Constraints Penalized Q-Learning (CPQ) treats OOD actions as unsafe and updates the policy using only safe state-action pairs [10]. Constraint-Conditioned Actor-Critic (CCAC) employs a constraint-conditioned variational autoencoder with a classifier to generate and identify unsafe OOD data, and uses such samples to regularize critics and policy learning [11]. Complementary to these, Lee et al. [12] proposed a method that optimizes the policy in the stationary distribution space under conservative cost constraints. Variational Optimization with Conservative Estimation (VOCE) utilizes variational formulations with pessimistic reward and cost value estimation to reduce OOD extrapolation errors [13]. Despite these advances, most methods assume that the dataset provides sufficiently informative cost supervision. In practice, sparse non-zero costs violate this assumption, which biases cost value estimates and weakens constraint enforcement.

Imperfect datasets complicate safe offline RL because the data distribution can be dominated by unsafe trajectories. Recent work attempts to reshape the offline distribution by generating or augmenting data. Generative Trajectory Augmentation (GTA) augments trajectories by diffusion-based denoising with guidance toward amplified returns, producing high-reward data [14]. AdaptDiffuser generates expert data with reward-gradient guidance and selects high-quality data via a discriminator, iteratively finetunes the diffusion planner [15]. For offline safe RL, trajectory classification partitions trajectories into desirable and undesirable subsets, training a policy to generate desirable trajectories via classifier-provided desirability scores [16]. OASIS, short for cOnditionAl diStributIon Shaping, employs a conditional diffusion model, conditioning on reward and cost thresholds to reshape the offline distribution toward safer and more rewarding regions [17]. SafeDiffuser embeds control barrier function constraints into the denoising process to enforce safety specifications during diffusion-based data generation [18]. However, distribution debiasing via data generation can be limited under severe dataset imbalance, since the generator is difficult to train and may fail to reliably produce safe and informative samples.

Due to their ability to represent complex distributions, diffusion models have been explored for offline decision-making by modeling policies as action generators [19]. In robotics, Chi et al. [20] generate robot behavior by modeling visuomotor control as a conditional denoising diffusion process. Several methods further incorporate safety into diffusion-based offline policies. Trajectory-based REal-time Budget Inference (TREBI) transforms policy optimization into a trajectory distribution optimization problem, using diffusion-based planning with dynamic cost budgets to guide action generation [21]. FeasIbility-guided Safe Offline RL (FISOR) leverages reachability analysis to translate hard safety requirements into feasible-region identification and derives an energy-guided diffusion formulation for weighted behavior cloning [22]. Constrained Diffusion Policy (CDP) maps diffusion samples onto a constrained manifold via a mirror diffusion model, thereby generating actions that satisfy safety constraints [23]. In safety-critical autonomous driving, Uncertainty-based Alternative Diffusion Policy (UADP) trains two alternative diffusion policies with an ensemble Q critic and selects actions with lower uncertainty to reduce risk [24]. However, many diffusion-based methods are trained by matching the offline data distribution and are thus sensitive to dataset quality. Reliable improvement beyond the behavior policy is challenging, especially under sparse cost supervision and severe dataset imbalance.

To tackle these challenges, we propose ARMOR (multi-sc**A**le **R**eweighting with **M**ulti-task **O**ffline c**R**itic), an actor-critic method that integrates multi-scale distribution debiasing and a multi-task critic into a conditional diffusion policy. ARMOR provides a generative offline control approach that jointly optimizes task performance and safety. The main contributions are summarized as follows:

- We propose a multi-task critic with a shared trunk that treats reward, long-term cost, and short-term cost as multiple tasks. The shared trunk learns shared representations to capture common state features, leveraging dense reward feedback to enable more reliable value estimation under sparse non-zero cost supervision.
- We present a multi-scale debiasing strategy that combines trajectory-level weighting with counterfactual transition-level weighting to mitigate dataset imbalance. At the trajectory level, we upweight low-cost trajectories. At the transition level, we assign transition weights by comparing short-term costs under counterfactual action perturbations, emphasizing safety-critical transitions near the risk boundary.
- We incorporate the above designs into a conditional diffusion actor. The actor is trained with weighted behavior cloning, augmented with a return-guided objective from the reward critic and a cost constraint from the long-term cost critic to achieve a better reward-cost trade-off.
- We evaluate ARMOR on eight tasks across two standard robot control benchmarks. Results show that ARMOR improves returns while satisfying cost limits and exhibits zero-shot adaptation.

2 Problem Formulation

We model safe reinforcement learning for continuous-control robotics using a Constrained Markov Decision Process (CMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, c, \gamma)$, where $\mathcal{S}$ is the state space, including variables like velocities, body poses, and task-relevant quantities such as target and obstacle positions. The action space $\mathcal{A}$ is typically continuous and corresponds to control commands, including torques, angular velocities, or acceleration. The transition function $P(s_{t+1} \mid s_t, a_t)$ represents the environment dynamics, determined by physics and contact interactions. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ quantifies task accomplishment, such as forward progress, staying on a desired track, or reaching a goal. The cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ quantifies safety violations that are critical in robot deployment, such as exceeding speed limits, leaving the safe region, or collisions. we denote the realized reward and cost at time t as $r_t \triangleq r(s_t, a_t)$ and $c_t \triangleq c(s_t, a_t)$. $\gamma \in (0, 1)$ is a discount factor. The policy $\pi(a \mid s)$ maps the current state to a control action. Given a trajectory $\tau = \{s_t, a_t, r_t, c_t\}_{t=0}^{T}$, the reward return and the cost return are defined as $R(\tau) = \sum_{t=0}^{T} (\gamma_r)^t r_t$ and $C(\tau) = \sum_{t=0}^{T} (\gamma_c)^t c_t$, where we allow different discount factors for reward and cost.

In safe robotic control, the policy is required to maximize the reward return while satisfying a cost constraint. Therefore, the CMDP objective can be written as:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi}[R(\tau)] \quad \text{s.t.} \quad \mathbb{E}_{\tau \sim \pi}[C(\tau)] \leq d, \tag{1}$$

where d is a prescribed cost limit.

Unlike the online setting, offline reinforcement learning assumes that the agent only has access to a fixed dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ collected by one or several behavior policies $\pi_B(a|s)$. Thus, the OSRL problem can be formulated as using an offline dataset $\mathcal{D}$ to learn a policy π that satisfies Eq. (1).

3 Method

In this section, we present ARMOR (multi-sc**A**le **R**eweighting with **M**ulti-task **O**ffline c**R**itic) as illustrated in Fig. 2.

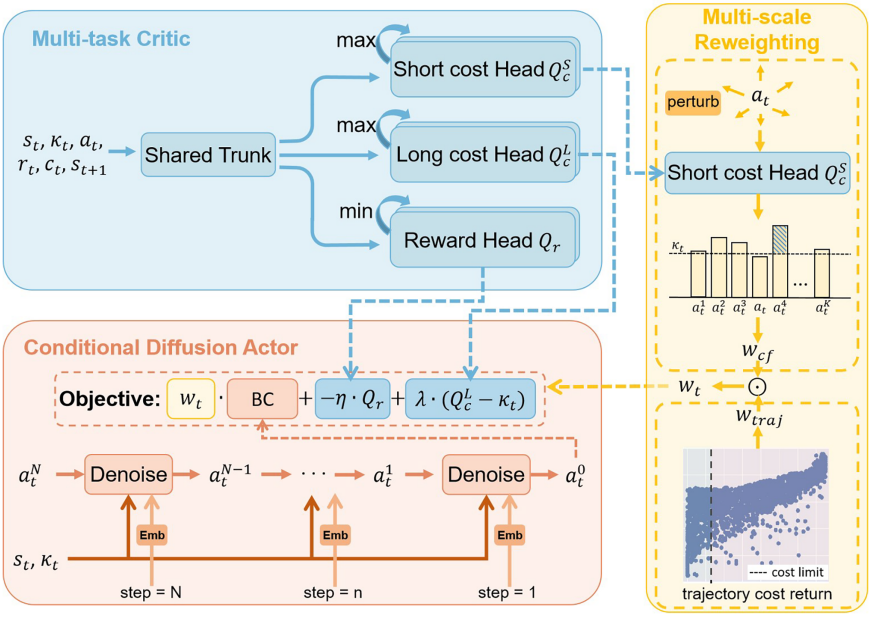

Figure 2: ARMOR overview.

In order to represent the remaining cost budget at each steps, we first introduce the cost threshold construction, where each transition is augmented with a cumulative cost. Then, we describe the proposed multi-task critic with a shared trunk, which jointly learns reward, long-term cost and short-term cost. The reward head guides performance optimization, the long-term cost head estimates cumulative cost to ensure policy safety, and the short-term cost head captures imminent violations to highlight the risk boundary. The shared trunk learns shared representations to ease risk representation learning, and PCGrad mitigates gradient conflicts across multi-task. Next, we formulate multi-scale reweighting, which combines trajectory-level debiasing with counterfactual transition-level weighting. Trajectory weights reshape the effective training distribution toward low-cost behaviors. Counterfactual comparisons use the short-term cost critic to identify transitions that are sensitive to small action perturbations. This focuses learning on safety-critical transitions near the constraint boundary. Finally, we integrate these components into a conditional diffusion actor, optimised through weighted behavior cloning, reward-guided improvement and a Lagrangian penalty induced by the long-term cost critic. This setup allows for stable offline training and a stronger reward-safety trade-off.

3.1 Cost Threshold Construction

Many previous methods typically distribute the cost limit across each time step either by discounting or uniformly splitting the total, and use this distribution to determine constraint violations at each time step. While this approach simplifies constraint evaluation, it introduces two main issues. First, safety cost signals are often sparse, with c_t being zero for most time steps. Distributing the cost limit uniformly across steps fails to align with the true cumulative risk process. Second, the CMDP constraint is inherently based on cumulative cost over a trajectory, and comparing step costs to step limits does not accurately reflect the cumulative constraint, leading to potential bias in constraint evaluation.

ARMOR introduces κ_t to represent the cost threshold from the current time step to the trajectory termination. Specifically, for any trajectory in the offline dataset, at time step t, we define the cumulative cost from time step t to the trajectory's end as the threshold for that step, i.e., $\kappa_t = \sum_{k=t}^{T} c_k$. κ_t dynamically changes

over time, enabling a more accurate representation of the remaining cost budget at different stages within a trajectory. This approach aligns more closely with the CMDP's cumulative cost constraint and provides consistent contextual information for critic and conditioned actor learning.

3.2 Multi-Task Critic

We transfer dense reward supervision to cost-related tasks via a shared representation, thereby reducing the negative impact of cost sparsity. This design eases risk representation learning and improves feature generalization for value estimation. Moreover, by sharing features across objectives, ARMOR reuses supervision more effectively, leading to more data-efficient value estimation in the offline setting.

Specifically, we formulate critic learning as multi-task representation learning. The critic takes (s_t, κ_t, a_t) as input and outputs three action-values:

$$\begin{aligned} Q_r(s_t, \kappa_t, a_t; \theta_r) &\approx \mathbb{E}\left[\sum_{k=0}^{T} (\gamma_r)^k r_{t+k} \mid s_t, \kappa_t, a_t, \pi\right], \\ Q_{c^u}(s_t, \kappa_t, a_t; \theta_{c^u}) &\approx \mathbb{E}\left[\sum_{k=0}^{T} (\gamma_{c^u})^k c_{t+k} \mid s_t, \kappa_t, a_t, \pi\right], \ u \in \{L, S\}. \end{aligned} \tag{2}$$

where Q_r is the reward critic that estimates the expected return, Q_{c^L} is the long-term cost critic that estimates the expected cost, and Q_{c^S} is the short-term cost critic that predicts imminent risk to identify safety-critical transitions. The discount factors γ_r, γ_{c^L}, and γ_{c^S} correspond to the reward, long-term cost, and short-term cost, respectively. $\gamma_{c^S} < \gamma_{c^L}$ encourages the short-term head to emphasize imminent risk.

The critic is implemented as a shared trunk $h = f_\psi(s, \kappa, a)$ followed by three task heads, each head maintains two Q networks [25]. Following conservative design for offline learning, we use the *minimum* reward estimate and the *maximum* cost estimate when constructing the Bellman targets. For a transition $(s_t, \kappa_t, a_t, r_t, c_t, s_{t+1}, \kappa_{t+1})$, let $a_{t+1} \sim \pi_\phi(\cdot \mid s_{t+1}, \kappa_{t+1})$ be an action sampled from the current policy. The targets are

$$\begin{aligned} y_r &= r_t + \gamma_r \min_{i\in\{1,2\}} Q_{r,i}(s_{t+1}, \kappa_{t+1}, a_{t+1}; \theta_r^-), \\ y_{c^u} &= c_t + \gamma_{c^u} \max_{i\in\{1,2\}} Q_{c^u,i}(s_{t+1}, \kappa_{t+1}, a_{t+1}; \theta_{c^u}^-), \ u \in \{L, S\}. \end{aligned} \tag{3}$$

θ^- denotes the target network. Each head is trained by a temporal difference (TD) loss:

$$\mathcal{L}_f = \mathbb{E}_{(s_t,\kappa_t,a_t)\sim\mathcal{D}}\left[\left(Q_f(s_t, \kappa_t, a_t; \theta_f) - y_f\right)^2\right], \ f \in \{r, c^L, c^S\}. \tag{4}$$

We jointly optimize all critic parameters $\theta = (\psi, \theta_r, \theta_{c^L}, \theta_{c^S})$. Learning multiple tasks simultaneously may lead to gradient conflicts, and we empirically observe such conflicts during training (Appendix A). To address the gradient conflicts across objectives, we apply PCGrad [26] to the gradients on the shared trunk parameters. Specifically, we perform backpropagation for each task loss $\mathcal{L}_i$ separately to compute the corresponding gradients $\mathbf{g}_i$. Then, for each head i, we calculate the gradient inner product $\mathbf{g}_i \cdot \mathbf{g}_j$ with other heads j. If the inner product is positive, the gradient remains unchanged. If the inner product is negative, we project $\mathbf{g}_i$ onto the orthogonal space of $\mathbf{g}_j$ (Fig. 3):

$$\mathbf{g}_i \leftarrow \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \mathbf{g}_j. \tag{5}$$

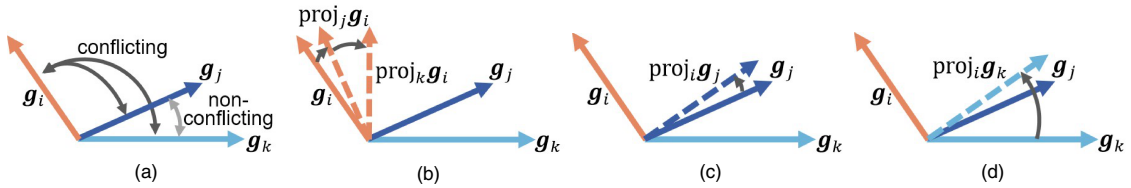

Figure 3: Multi-task gradients and PCGrad. (**a**) Gradient conflicts arise between tasks i and j as well as i and k, while the gradients of tasks j and k do not conflict; (**b**–**d**) Illustrate the PCGrad for each task.

Intuitively, when the gradients of two heads conflict, we subtract the conflicting component from the original gradient. After completing the gradient projection for all heads, we average the corrected gradients to obtain the joint gradient used for updating the shared trunk.

3.3 Multi-Scale Reweighting

Offline safe reinforcement learning is often limited by data distribution. Under strict constraints, few trajectories satisfy the safety requirements, resulting in an imbalanced training distribution. ARMOR addresses this issue by reshaping the training distribution with multi-scale weights, including trajectory-level weighting to correct behavioral bias and transition-level weighting to emphasize safe transitions near the constraint boundary.

Trajectory-level debiasing. For each trajectory $\tau_i \in \mathcal{D}$, we compute its cumulative cost $C_i = \sum_{t\in\tau_i} c_t$ and assign

$$w_i^{\text{traj}} = 1 + \beta \cdot \mathbb{I}[C_i \le d], \tag{6}$$

where $\beta > 0$ controls how strongly the training emphasizes low-cost trajectories, and $\mathbb{I}[\cdot]$ denotes the indicator function that equals 1 if the condition in the brackets is satisfied. All transitions within τ_i inherit the same w_i^{traj}, which biases learning toward feasible behaviors without discarding unsafe trajectories, thus preserving diversity and coverage.

The trajectory weight can be regarded as an importance shift toward feasible trajectories. Let $\nu(\tau)$ denote the dataset distribution over trajectories in $\mathcal{D}$. Define the feasible set $F = \{\tau : C(\tau) \le d\}$, and let $p = \nu(F)$, where $p \ll 1$ typically holds under data imbalance. With trajectory-level debiasing, define the weight function $w^{\text{traj}}(\tau) = 1 + \beta\,\mathbb{I}[\tau \in F]$ and the induced reweighted distribution $\nu_\beta(\tau) = \big(w^{\text{traj}}(\tau)\nu(\tau)\big)/\mathbb{E}_{\tau\sim\nu}[w^{\text{traj}}(\tau)]$. Then the probability mass of feasible trajectories under ν_β is $\nu_\beta(F) = \big((1+\beta)p\big)/\big((1+\beta)p + (1-p)\big)$, which is strictly increasing in β for $0 < p < 1$. Moreover, letting $\bar{C}_F = \mathbb{E}_{\tau\sim\nu}[C(\tau) \mid \tau \in F]$ and $\bar{C}_U = \mathbb{E}_{\tau\sim\nu}[C(\tau) \mid \tau \notin F]$, we obtain

$$\mathbb{E}_{\tau\sim\nu_\beta}[C(\tau)] = \frac{(1+\beta)p\,\bar{C}_F + (1-p)\bar{C}_U}{(1+\beta)p + (1-p)}, \quad \frac{d}{d\beta}\mathbb{E}_{\tau\sim\nu_\beta}[C(\tau)] = \frac{p(1-p)(\bar{C}_F - \bar{C}_U)}{\big((1+\beta)p + (1-p)\big)^2}. \tag{7}$$

Since $\bar{C}_F < \bar{C}_U$, increasing β strictly decreases $\mathbb{E}_{\tau\sim\nu_\beta}[C(\tau)]$, i.e., Eq. (6) performs an importance shift toward safe behaviors.

Counterfactual transition reweighting. It is not possible to judge the quality of all transitions based only on trajectory cost, since high-cost trajectories may also contain critical decision transitions. We therefore introduce a transition-level counterfactual weight that highlights critical safe transitions near the risk boundary, where the observed action is safe, but small local perturbations could lead to constraint violations.

For each transition (s_t, a_t, κ_t), we sample K perturbed actions $a_t^{(k)} = \text{clip}(a_t + \epsilon^{(k)}, a_{\min}, a_{\max})$ with $\epsilon^{(k)} \sim \mathcal{N}(0, \sigma^2 I)$, and evaluate them using the short-term cost critic. We define a counterfactual safety margin

$$m_t = \left[\max_{k=1,\ldots,K} Q_{c^s}(s_t, \kappa_t, a_t^{(k)}) - \kappa_t \right]_+ \cdot \mathbb{I}\left[Q_{c^s}(s_t, \kappa_t, a_t) \le \kappa_t \right], \tag{8}$$

where $[\cdot]_+$ denotes the positive-part operator. This margin becomes large when the current action is predicted safe but nearby actions can be unsafe. We then map m_t to a bounded weight via normalization and clipping:

$$w_t^{\text{cf}} = \min\left(1 + \frac{m_t}{\mathbb{E}[m] + \varepsilon}, w_{\max}\right), \tag{9}$$

where ε is a constant used to avoid division by zero. $\mathbb{E}[m]$ is estimated as the mean safety margin over a mini-batch, i.e., $\mathbb{E}[m] \approx \frac{1}{|B|} \sum_{t \in B} m_t$, where B is a mini-batch of transitions sampled from the dataset $\mathcal{D}$. Intuitively, for an action that is considered safe, if a small perturbation leads to a constraint violation, it indicates that the transition is near the risk boundary and the current action has successfully avoided the risk. In this case, we assign a larger value to w_t^{cf}, guiding the algorithm to focus on learning state-action pairs that are near the boundary but still safe.

The final weight is obtained by combining the two scales after normalization and clipping:

$$w_t = \text{clip}\left(\frac{w_i^{\text{traj}} w_t^{\text{cf}}}{\mathbb{E}[w_i^{\text{traj}} w_t^{\text{cf}}] + \varepsilon}, w_{\min}, w_{\max} \right), t \in \tau_i. \tag{10}$$

These weights are used to reweight the actor's behavior cloning loss, explicitly injecting safety-aware data preference from the distribution side rather than only relying on penalties in the objective.

Since w_t^{cf} relies on an evolving cost critic, we enable multi-scale reweighting after a warmup phase and periodically refresh the weights, while applying exponential moving average (EMA) smoothing to reduce weight oscillations.

3.4 Conditional Diffusion Actor

Diffusion models exhibit strong expressive capacity for modeling complex data distributions. ARMOR parameterizes the actor as a conditional diffusion policy. This formulation provides a unified framework for weighted behavior cloning, policy improvement based on the critic, and explicit safety constraint. Specifically, the policy generates actions a_t conditioned on (s_t, κ_t).

We follow the DDPM [27] on the action space. Let a_t^0 denote the dataset action and define a forward noising process

$$q(a_t^n \mid a_t^0) = \mathcal{N}\left(a_t^n; \sqrt{\bar{\alpha}_n}\, a_t^0, (1 - \bar{\alpha}_n) I\right), \tag{11}$$

where $n = 1, \ldots, N$, $\{\alpha_n\}_{n=1}^N$ is the noise schedule, and $\bar{\alpha}_n = \prod_{i=1}^n \alpha_i$. The denoising process is parameterized by a conditional distribution:

$$p_\phi(a_t^{n-1} \mid a_t^n, s_t, \kappa_t) = \mathcal{N}\left(a_t^{n-1}; \mu_\phi(a_t^n, s_t, \kappa_t, n),\ \Sigma_\phi(a_t^n, s_t, \kappa_t, n)\right), \tag{12}$$

where ϕ represents the parameters of the policy network. On the offline dataset $\mathcal{D}$, the diffusion policy learns the behavior distribution through noise regression, attempting to recover the actions as accurately as possible

given (s_t, κ_t). Let $\hat{\epsilon}_\phi(a_t^n, s_t, \kappa_t, n)$ be the network's predicted noise, and $\epsilon \sim \mathcal{N}(0, I)$ be the true noise. The weighted diffusion behavior cloning loss is given by

$$\mathcal{L}_{\mathrm{BC}}(\phi) = \mathbb{E}_{(s_t, a_t^0, \kappa_t)\sim\mathcal{D}}\left[w_t \left\|\hat{\epsilon}_\phi(a_t^n, s_t, \kappa_t, n) - \epsilon\right\|_2^2\right], \tag{13}$$

where w_t is the weights used to emphasize safe samples. This term ensures that the policy update does not deviate from the distribution supported by the offline data, providing a optimization foundation for subsequent value guidance and safety constraints.

Behavior cloning typically fails to produce a policy that outperforms the dataset. To address this, we introduce a policy improvement objective based on the reward value function. Specifically, we sample actions $a_t^\pi \sim \pi_\phi(\cdot \mid s_t, \kappa_t)$ from the current diffusion policy and use the reward critic $Q_r(s_t, \kappa_t, a_t^\pi)$ to drive the policy towards higher return actions.

At the same time, policy improvement must satisfy cost constraints. We use the long-term cost critic $Q_{c^L}(s_t, \kappa_t, a_t^\pi)$ to evaluate the expected cost of the action, with the cost threshold κ_t serving as the upper bound for the feasible domain.

Considering all of the above, the optimization of the actor can be formulated as a constrained problem:

$$\min_\phi \mathcal{L}_{\mathrm{BC}}(\phi) - \eta\, \mathbb{E}_{(s_t,\kappa_t)\sim\mathcal{D}, a_t^\pi\sim\pi_\phi}\left[Q_r(s_t, \kappa_t, a_t^\pi)\right] \quad \text{s.t.} \quad \mathbb{E}_{(s_t,\kappa_t)\sim\mathcal{D}, a_t^\pi\sim\pi_\phi}\left[Q_{c^L}(s_t, \kappa_t, a_t^\pi)\right] \le \kappa_t, \tag{14}$$

where $\eta > 0$ is the reward guidance coefficient. This formulation clearly captures the optimization logic of the proposed method by maximizing reward while maintaining diffusion-based behavior cloning constraints and ensuring cost feasibility. The constraint $Q_{c^L}(s_t, \kappa_t, a_t^\pi) \le \kappa_t$ can be interpreted as a practical surrogate for the original CMDP cost bound. The intuition is that Q_{c^L} estimates the future cumulative cost, while κ_t represents the remaining cost budget. Therefore, enforcing $Q_{c^L}(s_t, \kappa_t, a_t^\pi) \le \kappa_t$ is directly analogous to enforcing that the cumulative cost should remain below the cost limit. Under accurate cost estimation and sufficient data coverage, this surrogate is aligned with controlling the cumulative cost along the trajectory.

We transform this constrained problem into an unconstrained optimization by applying Lagrangian relaxation. By introducing the dual variable $\lambda(\kappa_t)$, we obtain the actor objective:

$$\mathcal{L}(\phi) = \mathcal{L}_{\mathrm{BC}}(\phi) - \eta\, \mathbb{E}\left[Q_r(s_t, \kappa_t, a_t^\pi)\right] + \mathbb{E}\left[\lambda(\kappa_t)\left(Q_{c^L}(s_t, \kappa_t, a_t^\pi) - \kappa_t\right)\right], \tag{15}$$

where $\lambda(\cdot)$ is parameterized by a small network. In this minimization objective, when the predicted cost exceeds the threshold, the penalty term pushes the policy away from high-risk actions. When the constraint is satisfied, the penalty weakens, allowing reward-driven policy improvement to take effect. The dual network is updated through approximate dual ascent, and its loss is given by:

$$\mathcal{L}_\lambda = -\mathbb{E}_{(s_t,\kappa_t)\sim\mathcal{D}, a_t^\pi\sim\pi_\phi}\left[\lambda(\kappa_t)\left(Q_{c^L}(s_t, \kappa_t, a_t^\pi) - \kappa_t\right)\right]. \tag{16}$$

Eqs. (15) and (16) define a standard primal–dual optimization of a Lagrangian relaxation. The actor minimizes a behavior-regularized objective with reward guidance and a Lagrangian penalty, while the dual variable is updated by approximate dual ascent. The multi-scale weights are normalized and clipped to prevent extreme gradient amplification. These design choices empirically stabilize training and are consistent with common convergence conditions for stochastic primal-dual methods such as bounded stochastic gradients and suitably chosen step sizes. We report the evolution of $\lambda(\kappa_t)$ across tasks in Appendix B Fig. A1, which empirically supports the stability of the primal-dual optimization.

3.5 Deployment of ARMOR on Robotic Systems

ARMOR trains a conditional diffusion model that is deployed on a robotic system to enable autonomous control. The robot acquires its current state s_t through sensors at each time step. The cost threshold κ is set to the predefined cost limit d initially, and is updated after each action based on the incurred cost. The robot uses the current state and cost threshold to determine the action. After executing the action, the robot updates its state and computes the new cost threshold, completing a feedback loop that allows for closed-loop control. Due to the design of ARMOR, this process ensures that the robot balances task performance with safety constraints.

4 Experiments

We evaluate ARMOR on the public benchmarks. In addition to the main comparison against representative baselines, we conduct an ablation study to quantify the contributions of the proposed multi-task critics and multi-scale reweighting, and perform sensitivity analyses on hyperparameters of the diffusion policy and the reweighting scheme. Furthermore, we demonstrate the zero-shot adaptation capability of ARMOR to different cost limits without retraining.

4.1 Environments

We conduct experiments on continuous-control robotic tasks using the public benchmarks `Bullet-Safety-Gym` [28] and `Safety-Gymnasium` [29], which are commonly used in previous works.

In `Bullet-Safety-Gym`, we focus on the `Run` task with three robot types: `Ant`, `Ball`, and `Drone`. In this task, the agent is rewarded for traversing a corridor between two boundaries at high speed, while crossing the boundaries or exceeding the velocity limit incurs penalties (Fig. 4a).

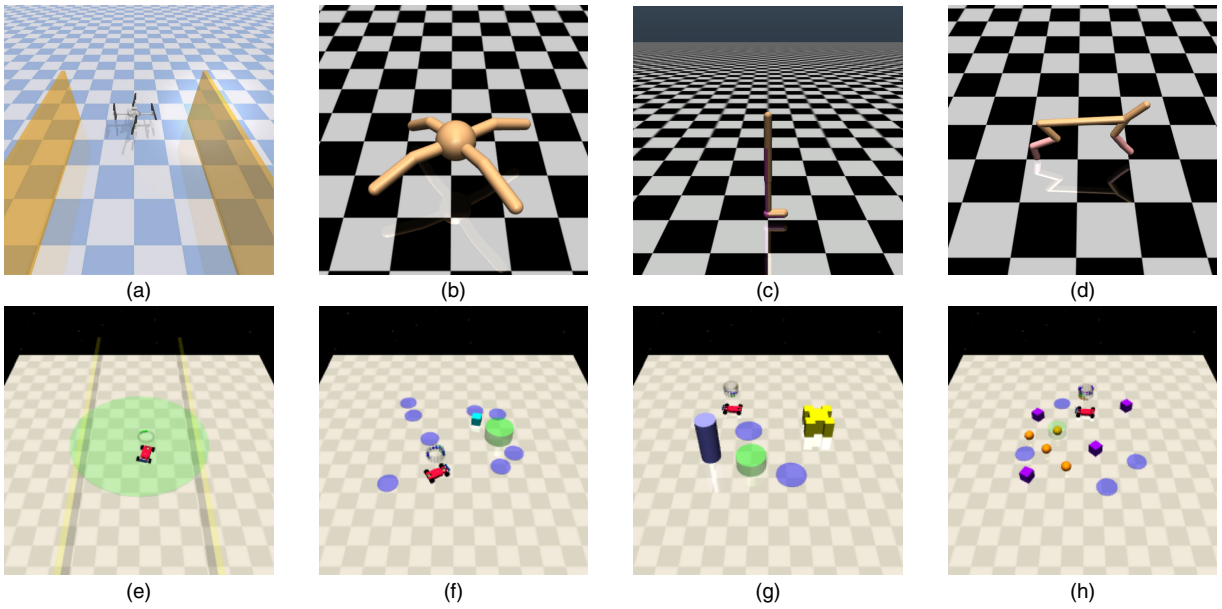

Figure 4: Tasks in `Bullet-Safety-Gym` and `Safety-Gymnasium`. (**a**) `Run`; (**b**) `AntVelocity`; (**c**) `Walker2dVelocity`; (**d**) `HalfCheetahVelocity`; (**e**) `Circle`; (**f**) `Goal`; (**g**) `Push`; (**h**) `Button`.

In Safety-Gymnasium, we consider two groups of tasks: the Velocity group and the Navigation group. In the Velocity group, the Ant, Walker2d, and HalfCheetah robots are used for tasks, where the objective is to maximize forward displacement and a cost is incurred when overspeed (Fig. 4b–d).

The Navigation group includes tasks Circle, Goal, Push, and Button with Point and Car robots (Fig. 4e–h). In Circle, clockwise motion along a circular track yields reward, whereas leaving the boundary-defined safe region produces a cost. In Goal, the agent navigates toward a target location while avoiding traps and preventing collisions with hazardous objects. In Push, the agent must push a box to the goal while steering around obstacles and avoiding traps. In Button, the agent must tap the correct target button among multiple buttons, and entering traps or collisions with moving obstacles trigger a cost.

4.2 Baselines

To provide a comprehensive evaluation, we compare ARMOR with the following offline baselines. This allows us to assess ARMOR from multiple perspectives, including whether it improves over behavior cloning, how it compares with representative Q-learning methods under sparse cost supervision, what additional benefits it brings within generative policy learning, and whether it can surpass data-generation approaches without relying on additional synthesized data.

- **Imitation Learning:** BC, behavior cloning that imitates trajectories in the datasets.
- **Q-Learning-Based Algorithms:** BCQL, a Lagrangian-based extension of BCQ [6]; CPQ [10], a Q-learning method that identifies Out Of Distribution (OOD) actions as unsafe and learns policies with safe transitions.
- **Generative Modeling Algorithm:** FISOR [22], a feasibility-guided method with a diffusion model.
- **Data Generation Algorithms:** OASIS [17], which employs a conditional diffusion model to generate datasets and guides the data distribution towards a target domain; CCAC [11], which generates and identifies unsafe OOD data to train adaptive safe policies.

4.3 Metrics

Our evaluation metrics include the normalized cost return and the normalized reward return.

$$R_{\text{normalized}} = \frac{R_\pi - R_{\min}}{R_{\max} - R_{\min}}, \quad C_{\text{normalized}} = \frac{C_\pi}{d}, \tag{17}$$

where R_π is the reward return of the policy π, $R_{\max}$ and $R_{\min}$ are the maximum and minimum reward returns within the given dataset, respectively. C_π is the cost return of the policy π, and d is the cost limit. A policy is safe if $C_{\text{normalized}} \leq 1$, and we pursue a higher reward return under this constraint.

According to difficulty, the cost limit is set to 5 for all Bullet-Safety-Gym tasks. For Safety-Gymnasium, Velocity tasks have their cost limits set to 10, and other tasks are set to 20. Table 2 lists the key hyperparameters. For a fair comparison, the common hyperparameters are kept consistent across methods, while method-specific hyperparameters are set according to the official implementations. More implementation details can be found in Appendix B.

Table 2: Key hyperparameters for ARMOR.

Hyperparameter	Value	Hyperparameter	Value
Training steps	200,000	Reweighting warmup steps	20,000

(Continued)

Table 2 (continued)

Hyperparameter	Value	Hyperparameter	Value
Diffusion denoising steps N	20	Reweighting EMA coefficient	0.3
Actor learning rate	1×10^{-4}	Reweighting interval	5000
Critic learning rate	1×10^{-3}	Trajectory weight control coefficient β	0.5
Batch size	512	Counterfactual samples K	8
Discount factors $(\gamma_r, \gamma_{c^L}, \gamma_{c^S})$	(0.99, 0.99, 0.3)	Counterfactual noise std σ	0.1
Reward guidance coefficient η	1	Reweighting clip range $[w_{\min}, w_{\max}]$	[0.8, 2.0]

4.4 Overall Performance

Table 3 reports the normalized return and normalized cost of all methods across three task groups.

Table 3: Evaluation results of the normalized reward and cost. Each value is averaged over 20 evaluation episodes and 3 seeds. Gray: Unsafe agents. Black: Safe agents. Blue: Safe agents with the highest reward.

Tasks	Metric	BC	BCQL	CPQ	OASIS	FISOR	CCAC	ARMOR
Ant-Run	reward	0.67 ± 0.04	0.54 ± 0.09	0.07 ± 0.02	0.24 ± 0.03	0.17 ± 0.01	0.12 ± 0.01	0.19 ± 0.03
	cost	7.40 ± 0.25	1.55 ± 0.79	0.01 ± 0.01	0.17 ± 0.09	0.00 ± 0.01	0.00 ± 0.00	0.17 ± 0.11
Ball-Run	reward	0.37 ± 0.16	0.21 ± 0.03	0.31 ± 0.01	0.30 ± 0.00	0.26 ± 0.01	0.31 ± 0.01	0.20 ± 0.04
	cost	5.41 ± 3.23	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.59 ± 0.99
Drone-Run	reward	0.60 ± 0.08	0.48 ± 0.08	0.42 ± 0.08	0.38 ± 0.03	0.09 ± 0.02	0.22 ± 0.22	0.53 ± 0.04
	cost	7.81 ± 6.54	8.55 ± 2.20	0.00 ± 0.00	2.42 ± 0.96	7.41 ± 1.31	4.29 ± 3.73	0.59 ± 0.58
Ant-Velocity	reward	0.96 ± 0.03	1.00 ± 0.01	−1.01 ± 0.00	0.94 ± 0.04	0.57 ± 0.03	−1.01 ± 0.01	0.71 ± 0.06
	cost	19.74 ± 4.62	15.77 ± 3.63	0.00 ± 0.00	4.24 ± 1.12	0.00 ± 0.00	0.00 ± 0.00	0.89 ± 0.11
Walker2d-Velocity	reward	0.73 ± 0.11	0.71 ± 0.01	0.12 ± 0.03	0.54 ± 0.33	0.07 ± 0.02	0.34 ± 0.24	0.75 ± 0.03
	cost	5.55 ± 1.81	0.03 ± 0.06	4.98 ± 2.06	0.88 ± 0.97	0.24 ± 0.10	6.04 ± 5.66	0.98 ± 0.69
HalfCheetah-Velocity	reward	0.94 ± 0.03	0.72 ± 0.12	0.30 ± 0.23	0.22 ± 0.09	0.65 ± 0.07	0.87 ± 0.01	0.20 ± 0.03
	cost	12.35 ± 9.20	0.06 ± 0.05	6.66 ± 7.35	0.00 ± 0.01	0.00 ± 0.00	0.94 ± 0.02	0.01 ± 0.02
PointCircle1	reward	0.77 ± 0.03	0.17 ± 0.07	0.70 ± 0.04	−0.18 ± 0.10	0.21 ± 0.05	0.53 ± 0.08	0.33 ± 0.10
	cost	7.91 ± 0.86	0.70 ± 0.39	10.61 ± 0.56	0.96 ± 1.03	12.03 ± 1.04	3.79 ± 0.81	0.55 ± 0.09
PointGoal1	reward	0.59 ± 0.05	0.64 ± 0.02	0.68 ± 0.18	0.43 ± 0.02	0.07 ± 0.03	0.76 ± 0.04	0.33 ± 0.07
	cost	1.96 ± 0.20	1.82 ± 0.10	2.95 ± 0.71	2.18 ± 0.52	0.26 ± 0.36	2.69 ± 0.39	0.87 ± 0.21
PointPush1	reward	0.19 ± 0.01	0.25 ± 0.04	0.06 ± 0.09	0.01 ± 0.01	0.10 ± 0.03	0.07 ± 0.22	0.17 ± 0.02
	cost	1.75 ± 1.42	1.25 ± 0.35	1.35 ± 1.36	0.59 ± 0.92	0.32 ± 0.20	1.30 ± 0.34	0.77 ± 0.34
PointButton1	reward	0.14 ± 0.06	0.21 ± 0.02	0.73 ± 0.04	0.05 ± 0.02	−0.03 ± 0.03	0.68 ± 0.04	0.07 ± 0.02
	cost	2.16 ± 0.85	3.38 ± 0.87	7.13 ± 0.55	4.19 ± 0.33	0.27 ± 0.19	6.88 ± 0.98	0.81 ± 0.07
CarCircle1	reward	0.72 ± 0.01	0.74 ± 0.02	0.74 ± 0.12	−0.42 ± 0.02	0.40 ± 0.02	0.23 ± 0.31	0.29 ± 0.08
	cost	9.82 ± 0.32	9.71 ± 1.03	10.51 ± 1.44	15.17 ± 3.02	1.51 ± 0.54	9.36 ± 6.87	0.84 ± 0.61
CarGoal1	reward	0.42 ± 0.07	0.42 ± 0.04	0.78 ± 0.04	−0.14 ± 0.07	−0.01 ± 0.01	0.82 ± 0.02	0.35 ± 0.05
	cost	1.19 ± 0.19	1.35 ± 0.04	2.01 ± 0.28	0.74 ± 0.38	0.00 ± 0.00	2.72 ± 0.05	0.73 ± 0.33

(Continued)

Table 3 (continued)

Tasks	Metric	BC	BCQL	CPQ	OASIS	FISOR	CCAC	ARMOR
CarPush1	reward	0.18 ± 0.04	0.22 ± 0.02	−0.16 ± 0.36	−0.92 ± 0.15	0.11 ± 0.02	−0.16 ± 0.48	0.20 ± 0.03
	cost	0.59 ± 0.31	1.18 ± 0.14	1.58 ± 0.59	0.02 ± 0.03	0.18 ± 0.10	0.91 ± 0.60	0.72 ± 0.22
CarButton1	reward	−0.04 ± 0.03	0.07 ± 0.01	0.40 ± 0.07	−0.30 ± 0.06	−0.17 ± 0.10	0.38 ± 0.05	0.09 ± 0.02
	cost	2.11 ± 1.18	1.78 ± 0.34	19.57 ± 2.67	1.97 ± 0.38	0.21 ± 0.14	19.56 ± 2.80	1.68 ± 0.13

Overall, ARMOR satisfies the cost constraint in most environments while attaining competitive returns. Furthermore, it demonstrates a distinct advantage in more challenging scenarios, where the environment is more complex. This outcome is primarily attributed to the reweighting mechanism, which reshapes the training distribution to emphasize safer and more informative transitions, aligning critic evaluation and diffusion policy learning. In addition, the shared representation provides a common feature basis for multiple critic heads, reducing reliance on sparse cost signals and improving the stability and accuracy of value estimation. For tasks with explicit goals, we further report success rate alongside normalized cost in Appendix C (Table A2).

BCQL employs a Lagrangian approach to balance performance and safety, whereas CPQ adopts a conservative update rule and updates the value function only on state action pairs classified as safe. However, neither method explicitly addresses distribution bias in offline data. When the proportion of safe samples in the dataset is low, the available supervisory signal becomes insufficient, which hinders learning policies that satisfy safety constraints while maintaining performance. OASIS synthesizes training data using reward and cost models, an inverse dynamics model, and a conditional diffusion generator. Since this pipeline involves multiple learned components, modeling errors introduced at any stage can propagate through subsequent data generation and policy optimization, which can degrade the resulting policy and increase the likelihood of constraint violations. CCAC updates the cost critic conservatively using augmented data to improve the reliability of constraint estimation, but it does not account for the potential sparsity of cost signals. When cost is sparse, the cost critic may fail to converge to an accurate estimate, which weakens constraint guidance during policy optimization and limits the ability to provide consistent safety guarantees. FISOR tends to enforce feasibility more strictly, yet in some tasks it yields very low even negative normalized return, suggesting that it may converge to overly conservative policies that are undesirable in offline safe RL.

We observe that ARMOR does not satisfy the cost constraint on `CarButton1`. Several baselines also violate the constraint on this environment, indicating that the dataset coverage and the sharp constraint boundary make it challenging for offline methods. While FISOR satisfies the constraint on `CarButton1`, its normalized return is negative, implying an extremely conservative policy. This behavior is not aligned with the goal of offline safe RL, which seeks both feasibility and high utility.

4.5 Ablation Study

We conduct ablation experiments to evaluate the effectiveness of each component in ARMOR. We consider the following variants: (i) **w/o Reweighting:** removing both trajectory-level and counterfactual reweighting by setting $w_t \equiv 1$; (ii) **w/o multi-critic:** replacing the multi-task critic with three independent critics and disabling PCGrad; (iii) **w/o multi-critic+reweighting:** keeping only the conditional diffusion actor and removing both the multi-task critic and reweighting; (iv) **w/o diffusion:** replacing the diffusion actor with a simple Gaussian policy; (v) **w/o warmup+EMA:** disabling the reweighting warmup and the exponential moving average used to smooth counterfactual weights; (vi) **w/o PCGrad:** removing PCGrad

in the shared critic trunk; (vii) **w/o multi-cost-critic:** replacing the multi-term cost critic with a single-head cost critic that has multi-horizon targets; (viii) **ARMOR:** the full method.

Fig. 5 summarizes the ablation results on `Navigation` tasks. Since offline safe RL prioritizes constraint satisfaction before return maximization, we primarily analyze the normalized cost. ARMOR achieves the lowest cost on three of four tasks and maintains competitive returns. Removing multi-scale reweighting leads to higher normalized cost, indicating that reshaping the training distribution is crucial for safety under offline data limitations. Ablating the multi-task critic also harms feasibility, suggesting that jointly learning reward and multi-horizon costs with conflict-aware optimization produces more reliable value estimates for policy learning. Removing both the multi-task critic and reweighting results in poorer performance in constraint satisfaction, highlighting that conditional generation alone is insufficient. Replacing the diffusion actor with a simple Gaussian policy causes constraint violation in most tasks, showing that the representation capacity of the diffusion model is indispensable. Disabling the reweighting warm-up and EMA tends to cost increases due to inaccurate value estimation during the initial training phase and unstable weights. Removing PCGrad and multi-term cost critic degrades constraint satisfaction, demonstrating the need to mitigate gradient interference in the shared critic and to capture both long-term and short-term safety signals.

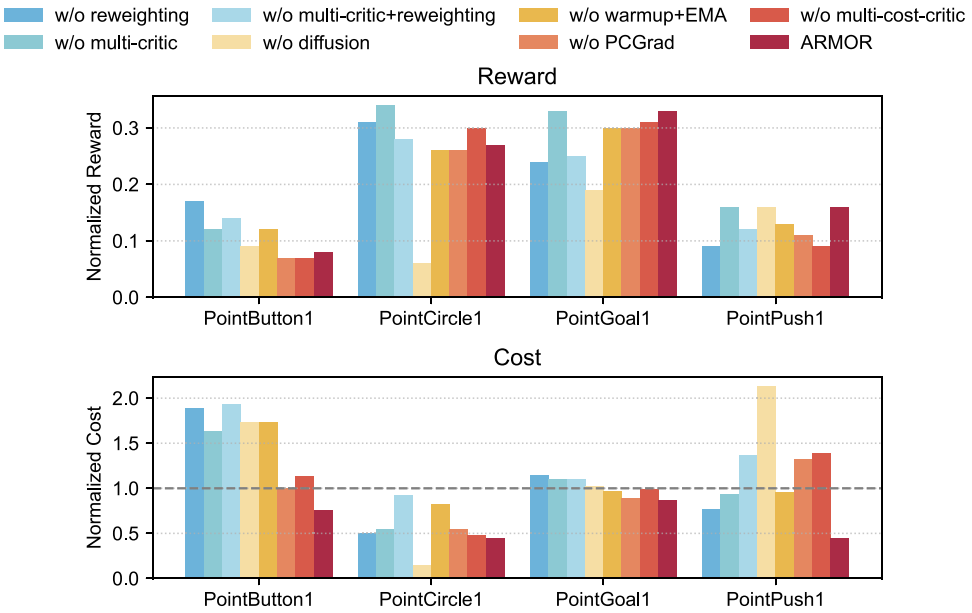

Figure 5: Ablations on `Navigation` tasks. The dashed line represents the normalized cost limit.

4.6 Hyperparameter Choices

We study the sensitivity of ARMOR to three important hyperparameters: the diffusion denoising steps N, the clipping range $[w_{\min}, w_{\max}]$ used in multi-scale reweighting and the discount factor γ_{c^S} for the short-term cost critic.

The denoising steps N control the granularity of the reverse diffusion process and thus affect both the expressiveness of the action generator and the inference cost. We sweep $N \in \{10, 20, 30\}$ and plot the training curves of normalized return and normalized cost in Fig. 6. We observe that larger N typically yields smoother curves with reduced variance, consistent with the intuition that finer denoising improves sampling stability. However, larger N increases inference time. Considering the trade-off between performance and runtime, we use $N = 20$ as the default setting in all main experiments.

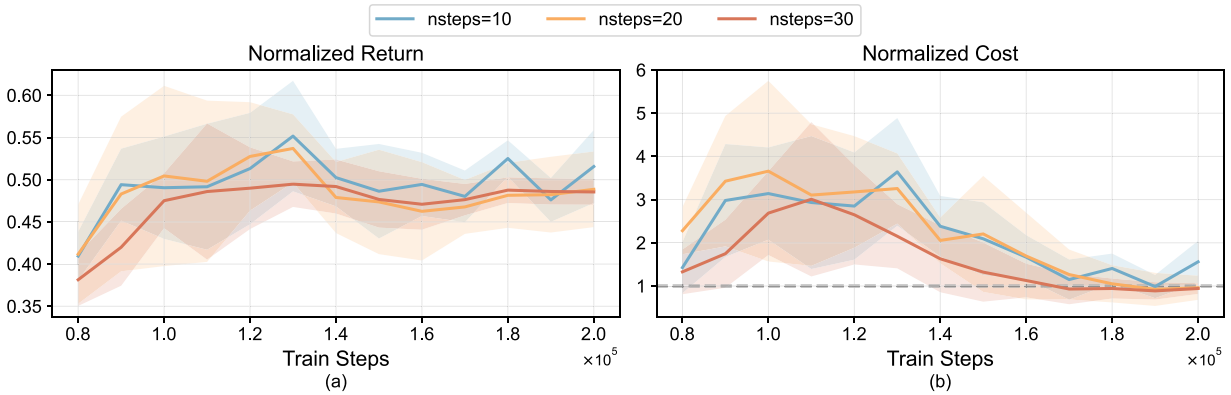

Figure 6: Effect of diffusion denoising step N during training. We report mean ± std across 3 seeds for (**a**) normalized return and (**b**) normalized cost.

We evaluate three clipping ranges for weights: $[0.9, 1.5]$, $[0.8, 2.0]$, and $[0.6, 2.5]$. A smaller lower bound suppresses gradients from low-confidence transitions more aggressively, but may reduce effective coverage and harm generalization. A larger upper bound emphasizes high-weight transitions, but may increase training instability and overfitting risk. Fig. 7 shows that the three ranges lead to broadly similar learning curves, suggesting that ARMOR is not overly sensitive to moderate clipping changes. We further demonstrated the performance of different clipping ranges during evaluation (Table 4), and selected $[0.8, 2.0]$ as the robust default range.

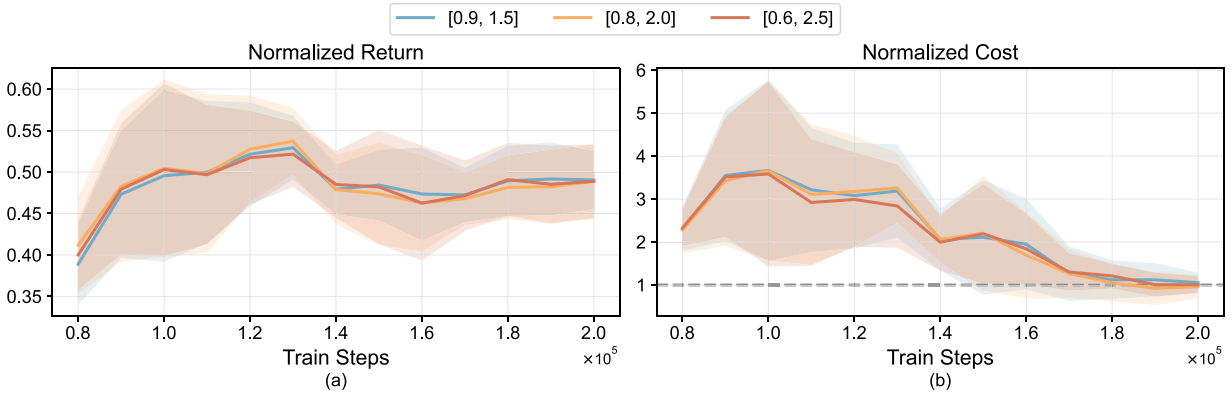

Figure 7: Effect of the weight clipping range $[w_{\min}, w_{\max}]$ during training. We report mean ± std across 3 seeds for (**a**) normalized return and (**b**) normalized cost.

Table 4: Evaluation results of clipping range.

Clipping Range	**[0.9, 1.5]**	**[0.8, 2.0]**	**[0.6, 2.5]**
reward	0.38 ± 0.10	0.33 ± 0.10	0.33 ± 0.09
cost	0.89 ± 0.27	0.55 ± 0.09	0.62 ± 0.12

The short-term cost discount γ_{c^s} sets the temporal scope of the short-term cost critic Q_{c^s}, and consequently controls how counterfactual reasoning identifies near-boundary transitions for reweighting. When γ_{c^s} is too small, the short-term cost target approaches to the immediate cost. Since costs are typically sparse

and the cost threshold κ_t aggregates future costs, the counterfactual margin $\max_{k=1,\ldots,K} Q_{c^S}(s_t, \kappa_t, a_t^{(k)}) - \kappa_t$ becomes positive less frequently, making the counterfactual weights degenerate toward 1. In contrast, when γ_{c^S} is too large, Q_{c^S} becomes less specialized for imminent risk, which blurs the risk boundary and weakens the emphasis on critical safe transitions. We sweep $\gamma_{c^S} \in \{0.1, 0.3, 0.5, 0.7\}$ and report the average normalized reward and normalized cost in Table 5. $\gamma_{c^S} = 0.3$ yields a strong reward-cost balance across tasks, achieving best reward while maintaining competitive normalized cost. Because of this, we use $\gamma_{c^S} = 0.3$ as the default setting.

Table 5: Evaluation results of different short-term cost discount factor. Gray: Unsafe agents. Black: Safe agents. Blue: Safe agents with the highest reward.

Task	0.1		0.3		0.5		0.7	
	Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Drone-Run	0.51	3.16	0.53	0.59	0.47	0.61	0.52	0.97
Ant-Velocity	0.65	0.74	0.71	0.89	0.58	0.73	0.63	0.76
PointCircle	0.28	0.57	0.33	0.55	0.29	0.54	0.26	0.55

4.7 Zero-Shot Adaptation

Another advantage of our approach is that it can adapt to different cost limits without retraining. In ARMOR, the cost limit is explicitly used when constructing the trajectory-level weights, so training with different limits induces policies with distinct safety-performance preferences. To examine generalization under changing constraints, we train two policies with cost limits of 15 and 30, and evaluate each policy under multiple limits $\{15, 20, 25, 30\}$, which cover both tightened and relaxed constraints. As shown in Fig. 8, ARMOR exhibits zero-shot adaptability when the constraint is relaxed. Even when the constraint is tightened, it maintains safety on most tasks, indicating robust safety control without fine-tuning. This performance arises from treating the dynamically updated constraint as an explicit input during training, which enables zero-shot adaptation to new cost limits. We further compare against CCAC trained with a cost limit of 15. While CCAC can generalize in some simpler tasks, it struggles on more challenging tasks and fails to achieve safe decision-making even under relaxed limits.

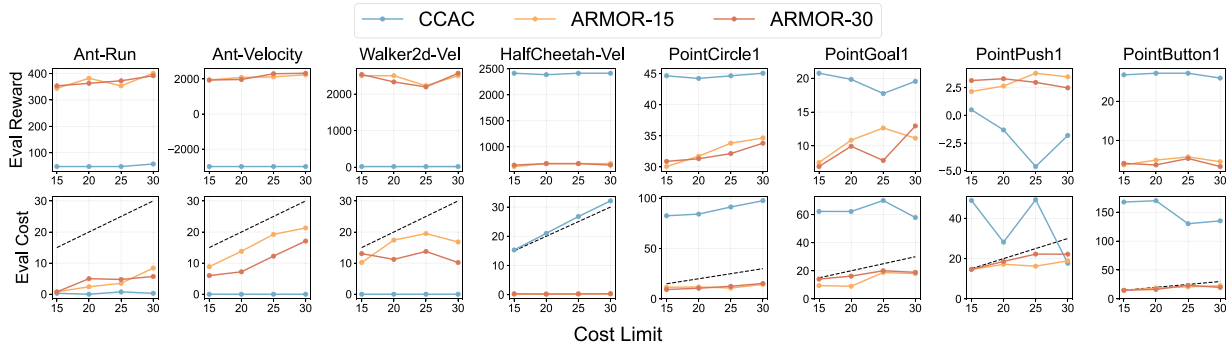

Figure 8: Zero-shot adaptation across cost limits. The dashed line represents the cost limit.

5 Conclusions

We propose ARMOR, a conditioned diffusion policy augmented with multi-scale reweighting and a multi-task critic. ARMOR learns a shared representation via the multi-task critic to enable reliable value

estimation. In addition, multi-scale reweighting is introduced to the conditional diffusion policy objective, which injects safety constraints from the data distribution. Experiments demonstrate that ARMOR achieves strong performance under cost constraints across multiple continuous-control robotics tasks. A practical limitation of ARMOR is its real-time inference overhead (Appendix C, Table A3), since diffusion-based action generation requires multi-step denoising for each decision. A further limitation is that the current study assumes accurate reward and cost signals, and does not consider noisy supervision that may arise in practical deployment. Future work could focus on accelerating inference and improving robustness to noisy feedback, which is essential for deploying ARMOR in real-world robotic systems.

Acknowledgement: The authors sincerely thank all those who supported to this research.

Funding Statement: This work was supported by the Joint Fund for Regional Innovation and Development of the National Natural Science Foundation of China (No. U22A20167) and the Special Project for Guiding the Transformation of Scientific and Technological Achievements in Shanxi Province (No. 202404021301033).

Author Contributions: The authors confirm contribution to the paper as follows: methodology, Chengjing Li; software, Chengjing Li; validation, Xiaoyan Zhao; investigation, Chengjing Li; data curation, Xiaoyan Zhao; writing—original draft preparation, Chengjing Li; writing—review and editing, Li Wang and Xiaoyan Zhao; visualization, Chengjing Li; supervision, Li Wang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Gradient Conflict Statistics

To examine whether gradient conflict is common, we measure the cosine similarity between the gradients induced by the three critic heads. At each training step, we compute cosine similarities for each of the three head pairs and count the number of pairs whose cosine similarity is negative. Table A1 reports the fraction of conflicting pairs for each task. The results indicate that gradient conflict is non-negligible. Gradient conflict occurs in roughly half of the training steps, and the case of two conflicting pairs arises frequently. This pattern is expected because the critic optimizes one reward objective and two cost objectives, while reward-driven gradients often compete with cost-driven gradients. Overall, these statistics support the motivation for applying PCGrad in the shared critic trunk to mitigate destructive interference among reward and multi-term cost learning.

Table A1: Gradient conflict statistics for PCGrad. Entries report the fraction of training steps with 0/1/2/3 critic-head pairs exhibiting negative gradient cosine similarity.

Tasks	0	1	2	3
PointCircle1	53.782%	22.137%	23.997%	0.085%
PointGoal1	49.634%	7.529%	42.832%	0.006%
PointPush1	49.619%	7.095%	43.280%	0.006%
PointButton1	49.373%	9.987%	40.631%	0.009%
Average	50.602%	11.687%	37.685%	0.026%

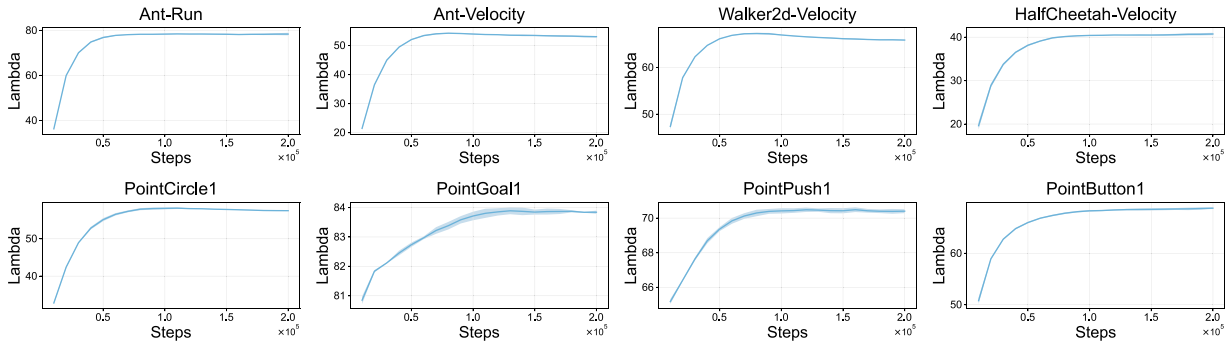

Figure A1: Evolution of the dual variable $\lambda(\kappa_t)$ during training.

Appendix B Implementation Details

The diffusion actor uses an MLP denoiser conditioned on state s_t, cost threshold κ_t, and diffusion timestep n. The cost condition embedding uses a 3-layer MLP, and the timestep uses sinusoidal positional embedding. The denoiser takes the concatenation of the noisy action, the state, and the two embeddings as input, then applies a 3-layer MLP trunk of width 256 with Mish activations, followed by a linear projection to the action dimension. The actor is optimized using Adam with learning rate 1×10^{-4}.

We employ a multi-task critic with a shared MLP trunk and three task-specific heads. The critic input concatenates the state, the cost threshold, and the action $[s_t; \kappa_t; a_t]$. The shared trunk is a 2-layer MLP with hidden sizes 256 and Mish activations. Each task head is a 2-layer MLP with hidden size 128 and Mish activations, outputting a scalar Q-value. We use a target critic to soft update at every training step: $\theta_{\text{targ}} \leftarrow \tau\theta + (1-\tau)\theta_{\text{targ}}$, where $\tau = 0.005$. The critic is optimized using Adam with learning rate 1×10^{-3}.

The dual variable is implemented as a MLP that takes the cost threshold κ_t as input. To ensure $\lambda(\kappa) \geq 0$, we treat the network output as $\log\lambda$ and compute $\lambda = \exp(\text{clip}(\log\lambda, -20, 5))$. The MLP has two hidden layers of width 256 with ReLU activations. We use Adam with learning rate 1×10^{-4}. The dual network is optimized jointly with the actor and is updated once per training step.

We report the evolution of the learned dual variable λ across tasks in Fig. A1. As described in Eqs. (15) and (16), λ increases when the predicted long-term cost exceeds the threshold and decreases otherwise, thereby adjusting the penalty strength. The curves quickly increase in early training and then stabilize, which empirically supports the numerical stability of the primal-dual optimization.

Appendix C More Experiment Results

For tasks with explicit goals, such as `Goal`, `Push`, and `Button`, we report success rate together with normalized cost to assess performance under safety constraints. Success is defined by the environment-provided success signal for reaching the goal, and the success rate is computed over 20 evaluation episodes. As shown in Table A2, all results are reported as the mean ± std across 3 seeds. Overall, the conclusions drawn from success rate are broadly consistent with those based on normalized return in Table 3. ARMOR achieves high success rates while satisfying the safety constraint on most tasks, demonstrating a favorable safety-performance trade-off.

Table A2: Evaluation results of success rate and normalized cost. Gray: Unsafe agents. Black: Safe agents. Blue: Safe agents with the highest success rate.

Tasks	Metric	BC	BCQL	CPQ	OASIS	FISOR	CCAC	ARMOR
PointGoal1	success rate	0.97 ± 0.06	0.97 ± 0.03	0.78 ± 0.18	0.97 ± 0.03	0.53 ± 0.15	1.00 ± 0.00	1.00 ± 0.00
	cost	1.96 ± 0.20	1.82 ± 0.10	2.95 ± 0.71	2.18 ± 0.52	0.26 ± 0.36	2.69 ± 0.39	0.87 ± 0.21
PointPush1	success rate	0.67 ± 0.03	0.62 ± 0.28	0.18 ± 0.18	0.00 ± 0.00	0.37 ± 0.12	0.47 ± 0.36	0.77 ± 0.08
	cost	1.75 ± 1.42	1.25 ± 0.35	1.35 ± 1.36	0.59 ± 0.92	0.32 ± 0.20	1.30 ± 0.34	0.77 ± 0.34
PointButton1	success rate	0.65 ± 0.05	0.90 ± 0.05	0.83 ± 0.14	0.57 ± 0.08	0.23 ± 0.03	1.00 ± 0.00	0.77 ± 0.16
	cost	2.16 ± 0.85	3.38 ± 0.87	7.13 ± 0.55	4.19 ± 0.33	0.27 ± 0.19	6.88 ± 0.98	0.81 ± 0.07
CarGoal1	success rate	0.88 ± 0.06	0.98 ± 0.03	0.60 ± 0.49	0.10 ± 0.17	0.20 ± 0.05	1.00 ± 0.00	0.93 ± 0.08
	cost	1.19 ± 0.19	1.35 ± 0.04	2.01 ± 0.28	0.74 ± 0.38	0.00 ± 0.00	2.72 ± 0.05	0.73 ± 0.33
CarPush1	success rate	0.63 ± 0.14	0.73 ± 0.03	0.12 ± 0.20	0.00 ± 0.00	0.38 ± 0.06	0.57 ± 0.15	0.68 ± 0.08
	cost	0.59 ± 0.31	1.18 ± 0.14	1.58 ± 0.59	0.02 ± 0.03	0.18 ± 0.10	0.91 ± 0.60	0.72 ± 0.22
CarButton1	success rate	0.78 ± 0.06	0.78 ± 0.13	0.48 ± 0.21	0.15 ± 0.05	0.10 ± 0.05	0.98 ± 0.03	0.83 ± 0.03
	cost	2.11 ± 1.18	1.78 ± 0.34	19.57 ± 2.67	1.97 ± 0.38	0.21 ± 0.14	19.56 ± 2.80	1.68 ± 0.13

Table A3 reports the inference time of ARMOR and Q-learning style offline safe RL baselines, measured on an NVIDIA RTX 3090 GPU. ARMOR exhibits the largest latency among the compared methods. This overhead is expected because the actor is a conditional diffusion policy and requires multiple denoising steps at test time. In contrast, Q-learning style baselines select actions with a single forward pass through the policy. Despite this overhead, the per-action latency is still on the order of 10^{-2} seconds, suggesting that ARMOR is still feasible for moderate control frequencies in real-world robotics.

Table A3: Per-action inference latency (s).

	BC	BCQL	CPQ	CCAC	ARMOR
Inference time	0.001473	0.001568	0.001704	0.001706	0.022674

References

1. Saeidi H, Opfermann JD, Kam M, Wei S, Léonard S, Hsieh MH, et al. Autonomous robotic laparoscopic surgery for intestinal anastomosis. Sci Robot. 2022;7(62):eabj2908. doi:10.1126/scirobotics.abj2908.
2. Wu J, Huang Y, Lai Y, Yang S, Zhang C. Obstacle avoidance inspection method of cable tunnel for quadruped robot based on particle swarm algorithm and neural network. Sci Rep. 2025;15(1):36065. doi:10.1038/s41598-025-19903-w.
3. Nie J, Zhang G, Lu X, Wang H, Sheng C, Sun L. Obstacle avoidance method based on reinforcement learning dual-layer decision model for AGV with visual perception. Control Eng Pract. 2024;153(8):106121. doi:10.1016/j.conengprac.2024.106121.
4. Radosavovic I, Xiao T, Zhang B, Darrell T, Malik J, Sreenath K. Real-world humanoid locomotion with reinforcement learning. Sci Robot. 2024;9(89):eadi9579. doi:10.1126/scirobotics.adi9579.
5. Liu Z, Guo Z, Lin H, Yao Y, Zhu J, Cen Z, et al. Datasets and benchmarks for offline safe reinforcement learning. J Data-Centric Mach Learn Res. 2024;1(12):1–29. doi:10.52202/079017-2494.
6. Fujimoto S, Meger D, Precup D. Off-policy deep reinforcement learning without exploration. In: Proceedings of the 36th International Conference on Machine Learning; 2019 Jun 9–15; Long Beach, California, USA. p. 2052–62.
7. Kumar A, Zhou A, Tucker G, Levine S. Conservative q-learning for offline reinforcement learning. In: Proceedings of the 34th International Conference on Neural Information Processing Systems; 2020 Dec 6–12; virtual. p. 1179–91.

8. Hong K, Li Y, Tewari A. A primal-dual-critic algorithm for offline constrained reinforcement learning. In: Proceedings of the 27th International Conference on Artificial Intelligence and Statistics; 2024 May 2–4; Palau de Congressos, Valencia, Spain. p. 280–8.
9. Polosky N, Da Silva BC, Fiterau M, Jagannath J. Constrained offline policy optimization. In: Proceedings of the 39th International Conference on Machine Learning; 2022 Jul 17-23; Baltimore, MD, USA. p. 17801–10.
10. Xu H, Zhan X, Zhu X. Constraints penalized q-learning for safe offline reinforcement learning. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence; 2022 Feb 22–Mar 1; virtual. p. 8753–60. doi:10.1609/aaai.v36i8.20855.
11. Guo Z, Zhou W, Wang S, Li W. Constraint-conditioned actor-critic for offline safe reinforcement learning. In: Proceedings of the 13th International Conference on Learning Representations; 2025 Apr 24–28; Singapore.
12. Lee J, Paduraru C, Mankowitz DJ, Heess N, Precup D, Kim KE, et al. COptiDICE: offline constrained reinforcement learning via stationary distribution correction estimation. In: Proceedings of the 10th International Conference on Learning Representations; 2022 Apr 25–29; virtual.
13. Guan J, Chen G, Ji J, Yang L, Zhou A, Li Z, et al. Voce: variational optimization with conservative estimation for offline safe reinforcement learning. In: Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. p. 33758–80.
14. Lee J, Yun S, Yun T, Park J. GTA: generative trajectory augmentation with guidance for offline reinforcement learning. In: Proceedings of the 38th International Conference on Neural Information Processing Systems; 2024 Dec 10–15; Vancouver, BC, Canada. p. 56766–801.
15. Liang Z, Mu Y, Ding M, Ni F, Tomizuka M, Luo P. AdaptDiffuser: Diffusion models as adaptive self-evolving planners. In: Proceedings of the 40th International Conference on Machine Learning; 2023 Jul 23-29; Honolulu, HI, USA. p. 20725–45.
16. Gong Z, Kumar A, Varakantham P. Offline safe reinforcement learning using trajectory classification. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence; 2025 Feb 25–Mar 4; Philadelphia, PA, USA. p. 16880–7. doi:10.1609/aaai.v39i16.33855.
17. Yao Y, Cen Z, Ding W, Lin H, Liu S, Zhang T, et al. OASIS: conditional distribution shaping for offline safe reinforcement learning. In: Proceedings of the 38th International Conference on Neural Information Processing Systems; 2024 Dec 10–15; Vancouver, BC, Canada. p. 78451–78. doi:10.52202/079017-2494.
18. Xiao W, Wang TH, Gan C, Hasani R, Lechner M, Rus D. Safediffuser: safe planning with diffusion probabilistic models. In: Proceedings of the 11th International Conference on Learning Representations; 2023 May 1–5; Kigali, Rwanda.
19. Ajay A, Du Y, Gupta A, Tenenbaum JB, Jaakkola TS, Agrawal P. Is conditional generative modeling all you need for decision-making? In: Proceedings of the 11th International Conference on Learning Representations; 2023 May 1–5; Kigali, Rwanda.
20. Chi C, Xu Z, Feng S, Cousineau E, Du Y, Burchfiel B, et al. Diffusion policy: visuomotor policy learning via action diffusion. Int J Robot Res. 2025;44(10–11):1684–704. doi:10.1177/02783649241273668.
21. Lin Q, Tang B, Wu Z, Yu C, Mao S, Xie Q, et al. Safe offline reinforcement learning with real-time budget constraints. In: Proceedings of the 40th International Conference on Machine Learning; 2023 Jul 23–29; Honolulu, HI, USA. p. 21127–52.
22. Zheng Y, Li J, Yu D, Yang Y, Li SE, Zhan X, et al. Safe offline reinforcement learning with feasibility-guided diffusion model. In: Proceedings of the 12th International Conference on Learning Representations; 2024 May 7–11; Vienna, Austria.
23. Ha T, Cha H, Ji D. CDP: constrained diffusion policies with mirror diffusion model for safety-assured imitation learning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems; 2025 Oct 19–25; Hangzhou, China. p. 9838–45. doi:10.1109/IROS60139.2025.11246518.
24. Huang X, Wang X, Cheng Y. Uncertainty-based alternative diffusion policy for safe autonomous driving. IEEE Trans Intell Transp Syst. 2025;26(11):18854–63. doi:10.1109/TITS.2025.3587341.
25. Hasselt H. Double Q-learning. In: Proceedings of the 24th International Conference on Neural Information Processing Systems; 2010 Dec 6–9; Vancouver, BC, Canada.

26. Yu T, Kumar S, Gupta A, Levine S, Hausman K, Finn C. Gradient surgery for multi-task learning. In: Proceedings of the 34th International Conference on Neural Information Processing Systems; 2020 Dec 6–12; virtual. p. 5824–36.
27. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Proceedings of the 34th International Conference on Neural Information Processing Systems; 2020 Dec 6–12; virtual. p. 6840–51.
28. Gronauer S. Bullet-Safety-Gym: a framework for constrained reinforcement learning. In: AAMAS '24: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent System. New York, NY, USA: The Association for Computing Machinery (ACM); 2022. doi:10.14459/2022md1639974.
29. Ji J, Zhou J, Zhang B, Dai J, Pan X, Sun R, et al. Omnisafe: an infrastructure for accelerating safe reinforcement learning research. J Mach Learn Res. 2024;25(285):1–6.