LEARNING ROBOTIC SKILLS FROM OFFLINE PREFERENCE-BASED RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Preference-based reinforcement learning (PbRL) trains agents from preference feedback without explicit task rewards. We introduce Preference-Score Actor-Critic (PSAC), which augments the critic update with a preference score learned directly from comparisons. Standard two-stage methods first fit a reward model and then apply an offline RL algorithm; errors in the proxy reward can propagate through value estimation. PSAC complements this reward signal with a separately learned state-desirability function and scores actions relative to the behavior distribution. PSAC achieves the highest mean score among the compared preference-based methods on 14 of 17 D4RL tasks, with ablations supporting the preference-score augmentation. In a MuJoCo simulation of Unitree G1 navigation over rough terrain, PSAC learns high-level control from 676 labeled preference pairs using a pre-trained locomotion controller. Code and videos: https://anonymous.4open.science/r/PSAC-6686.

1 INTRODUCTION

Reinforcement learning (RL) has advanced simulated control (Haarnoja et al., 2018), planning (Silver et al., 2008), and game playing (Silver et al., 2016). Extending these successes to robotics and human interaction requires reliable reward signals, which can demand complex instrumentation (Schenck & Fox, 2017; Peng et al., 2020) and labor-intensive calibration (Yu et al., 2020a). Reward design is especially difficult when a task combines multiple objectives (Faust et al., 2019).

Preference-based RL (PbRL) instead uses relative judgments about behavior, often collected through pairwise trajectory comparisons. Such feedback has supported learning in control (Wilson et al., 2012; Sadigh et al., 2017) and language applications (Stiennon et al., 2020). Offline PbRL combines these comparisons with pre-collected transitions, making it useful when trajectories are available but task rewards are difficult to specify.

Existing methods commonly learn a reward model from preference data and then apply an offline RL algorithm (Christiano et al., 2017; Kim et al., 2023). This pipeline relies on estimated rewards: approximation errors can propagate through bootstrapped value updates. Moreover, preferences constrain relative rankings rather than absolute utility. A learned scalar reward therefore introduces a choice of scale that can affect subsequent policy optimization.

We introduce **Preference-Score Actor-Critic (PSAC)**. Alongside a reward model based on the Preference Transformer (Kim et al., 2023), PSAC learns a state-desirability function directly from preferences using the Bradley–Terry model (Bradley & Terry, 1952). This auxiliary function is fitted without Bellman backups on the proxy reward. It assigns a preference score to an action by comparing its predicted next-state desirability with a baseline from the behavior distribution (Figure 1). The score then augments the critic update, providing a second signal grounded in preference rankings.

PSAC achieves the highest mean score among the compared preference-based methods on 14 of 17 D4RL tasks (Fu et al., 2020); Table 1 also isolates the contribution of preference-score augmentation. Further experiments compare direct and reward-based value learning (Section 5.2) and preference-data efficiency against PT (Section 5.3). Finally, PSAC learns high-level rough-terrain navigation in a Unitree G1 simulation from 676 labeled preference pairs, improving on behavior cloning (Section 5.4).

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

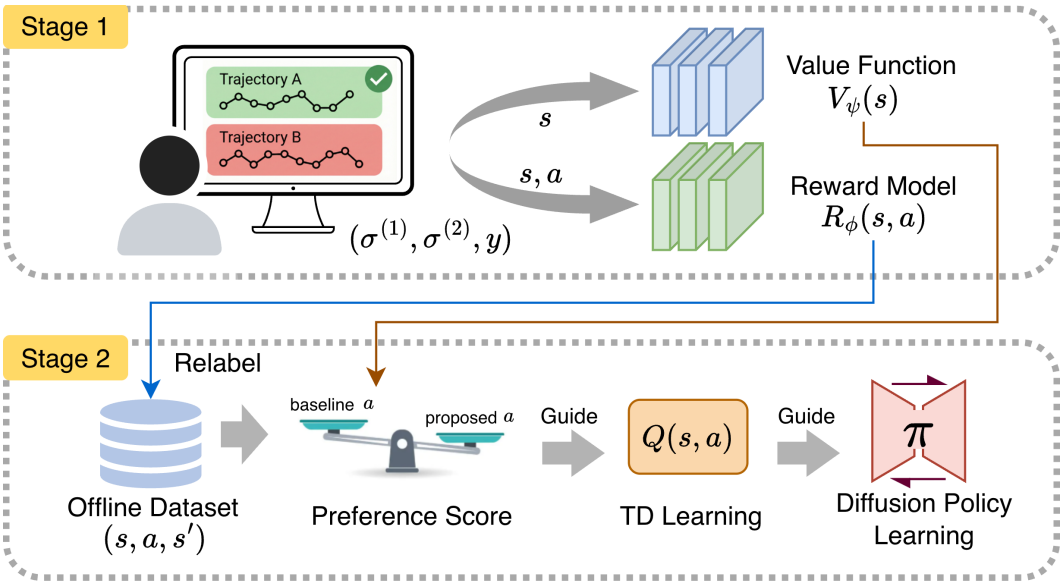


Figure 1: Overview of the proposed preference-score actor-critic framework. In Stage 1, the reward model and value function are learned from the preference dataset. In Stage 2, the preference score guides the offline learning of the diffusion policy.

Relation to prior work. PT (Kim et al., 2023) learns a reward model, whereas DPPO (An et al., 2023) and IPL (Hejna & Sadigh, 2023) optimize policies without a separate reward model. PSAC retains reward learning and adds a preference-derived signal to the critic. Appendix C reviews these connections.

2 Preliminaries

Preference-based Reinforcement Learning. We consider a discounted Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ represents the transition kernel, $\gamma \in (0, 1)$ is the discount factor, and ρ_0 corresponds to the initial-state distribution.

A policy $\pi(\cdot|\boldsymbol{s}) : \mathcal{S} \to \Delta(\mathcal{A})$ maps a given state $\boldsymbol{s}$ to a probability distribution over the action space. Let $\mu(\cdot \mid s)$ denote the behavior policy that generates the offline dataset, i.e., the trajectories $s_1 \sim \rho_0$, $a_t \sim \mu(\cdot \mid s_t)$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$.

PbRL uses preferences over trajectory segments as supervision. For a pair $(\sigma^{(1)}, \sigma^{(2)})$, each segment contains K state–action pairs. We use $y = 1$ when $\sigma^{(1)} \succ \sigma^{(2)}$, $y = 0$ for the reverse preference, and $y = 0.5$ for a tie, consistently with the losses below.

In this paper, we focus on *offline* PbRL, where learning is performed solely from an offline collection of preference annotations, $\mathcal{D}_p = \{(\sigma^{(1)}, \sigma^{(2)}, y)\}$, without any online interaction with the environment. This setting can turn static feedback datasets into effective decision-making policies, and is particularly attractive when online data collection is costly, risky, or otherwise impractical.

3 Method

3.1 Reward Learning

The standard paradigm for solving PbRL involves first learning a proxy reward function. A canonical approach to learning a reward model from preferences is grounded in the Bradley–Terry framework (Bradley & Terry, 1952). Given a preference dataset $\mathcal{D}_p = \{(\sigma^{(1)}, \sigma^{(2)}, y)\}$, each σ represents

a length-K trajectory segment:

$$\sigma = (s_1, a_1, s_2, a_2, \ldots, s_K, a_K),$$

where the label y indicates the preferred segment. While larger values of K increase the information content of each comparison, they may also introduce potential noise into the judgments. Under these annotations, the reward model is typically specified via the Bradley–Terry preference model (Bradley & Terry, 1952):

$$P_{r_E}\left[\sigma^{(1)} \succ \sigma^{(2)}\right] = \frac{\exp\left(\sum_t r_E\left(s_t^{(1)}, a_t^{(1)}\right)\right)}{\sum_{i\in\{1,2\}} \exp\left(\sum_t r_E\left(s_t^{(i)}, a_t^{(i)}\right)\right)}, \tag{1}$$

where $r_E(s_t, a_t)$ denotes the learned reward model. We use the subscript r_E on the probability P to indicate that the preference distribution stems from the expert's underlying reward function. To approximate r_E, prior works in PbRL estimate a parametric reward function r_ϕ by minimizing the binary cross-entropy over $\mathcal{D}_p$:

$$\begin{aligned}\mathcal{L}_R(\phi) = -\mathbb{E}_{(\sigma^{(1)},\sigma^{(2)},y)\sim\mathcal{D}_p}\Big[y \log P_{r_\phi}\big(\sigma^{(1)} \succ \sigma^{(2)}\big) \\ + (1-y)\log\Big(1 - P_{r_\phi}\big(\sigma^{(1)} \succ \sigma^{(2)}\big)\Big)\Big].\end{aligned} \tag{2}$$

Conventionally, a policy π can be updated using any standard RL algorithm to maximize the expected return with respect to this learned reward.

It is worth noting that Eq. (1) assumes the reward is Markovian (i.e., dependent solely on the current state and action) and that human evaluators assess trajectory quality based on an unweighted sum of rewards. However, this formulation has significant limitations in modeling realistic human preferences. First, specifying complex tasks via Markovian rewards is often challenging (Early et al., 2022). Furthermore, accurate credit assignment within a trajectory is crucial, as humans are often sensitive to specific salient moments rather than the uniform accumulation of rewards (Kahneman, 2000).

To address these limitations, we employ the Preference Transformer (Kim et al., 2023) to learn a reward function. This approach assumes that the probability of preferring a segment depends exponentially on a weighted sum of non-Markovian rewards:

$$\begin{aligned}&P_{r_\phi}\left[\sigma^{(1)} \succ \sigma^{(2)}\right] \\ &= \frac{\exp\Big(\sum_t w\big(\tau^{(1)};\phi\big)\cdot r\big(\tau_{1:t}^{(1)};\phi\big)\Big)}{\sum_{j\in\{1,2\}} \exp\Big(\sum_t w\big(\tau^{(j)};\phi\big)\cdot r\big(\tau_{1:t}^{(j)};\phi\big)\Big)},\end{aligned} \tag{3}$$

where $\tau^{(j)} = \{(s_i^j, a_i^j)\}_{i=1}^K$ denotes the full trajectory segment and $\tau_{1:t}^{(j)}$ denotes the sub-trajectory up to time t. To capture temporal dependencies, the model utilizes a non-Markovian reward function r, which receives the preceding sub-trajectory as input. Additionally, an importance weight w is introduced as a function of the entire segment, enabling the Preference Transformer to perform credit assignment within the segment. This formulation generalizes the conventional design; if the reward function depends only on the current state–action pair and the importance weight is uniformly set to 1, Eq. (3) reduces to the standard model in Eq. (1). We learn the reward model by minimizing the binary cross-entropy loss in Eq. (2), where the preference probability P_{r_ϕ} is parameterized as defined in Eq. (3).

3.2 Value Function Learning

A key innovation of our method is the direct learning of a value function from offline PbRL datasets, which allows for a dual exploitation of the preference data. For any trajectory pair where $\sigma^{(1)} \succ \sigma^{(2)}$, we aim to learn a value function $V_\psi(s)$ that upholds the ranking constraint: $\sum_t V_\psi(s_t^{(1)}) > \sum_t V_\psi(s_t^{(2)})$.

To achieve this, we formulate a binary classification objective grounded in the Bradley–Terry model. Specifically, for a trajectory pair $(\sigma^{(1)}, \sigma^{(2)})$ with the preference label $y = 1$, we define the preference

probability as:

$$P_{V_\psi}\left(\sigma^{(1)} \succ \sigma^{(2)}\right) = \frac{\exp\left(\sum_t V_\psi(s_t^{(1)})\right)}{\sum_{i\in\{1,2\}} \exp\left(\sum_t V_\psi(s_t^{(i)})\right)}. \tag{4}$$

The corresponding loss function is given by:

$$\begin{aligned}\mathcal{L}_V(\psi) = -\mathbb{E}_{(\sigma^{(1)},\sigma^{(2)},y)\sim\mathcal{D}_p}\Big[y \log P_{V_\psi}(\sigma^{(1)} \succ \sigma^{(2)}) \\ + (1-y)\log\Big(1 - P_{V_\psi}(\sigma^{(1)} \succ \sigma^{(2)})\Big)\Big].\end{aligned} \tag{5}$$

We interpret $V_\psi(s)$ as a preference-derived measure of *state desirability*. The model assumes that the sum of these scores predicts trajectory preference. This is a modeling choice rather than a Bellman characterization of discounted return.

Fitting V_ψ directly to preference labels avoids propagating reward-model errors through this auxiliary estimator. The actor–critic still uses a learned reward, so the construction does not eliminate all sources of reward or value approximation error.

To validate the superiority of this direct approach, we compare it against the conventional indirect method, where the value function is derived strictly from the learned proxy reward. In this standard baseline, the value function V_ψ is parameterized by a neural network and trained by minimizing the expectile regression loss (Kostrikov et al., 2021) over the offline dataset $\mathcal{D}_p$:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})\sim\mathcal{D}_p}\big[L_2^\tau(r_\phi + \gamma V_\psi(\boldsymbol{s'}) - V_\psi(\boldsymbol{s}))\big], \tag{6}$$

where r_ϕ is parameterized by Eq. (3) and learned via Eq. (2).

Section 5.2 compares the two estimators. Direct learning improves the reported average score, although it does not outperform indirect learning on every task. This comparison supports using a separately learned preference signal alongside the proxy reward.

3.3 Preference Score Learning

We then introduce the design of preference score. Let the one-step value predictor be defined as

$$m_\psi(s,a) \;\coloneqq\; \mathbb{E}_{s'\sim P(\cdot|s,a)}\big[V_\psi(s')\big]. \tag{7}$$

Consider the random variable $M_s \coloneqq m_\psi(s,A)$ induced by the behavior policy $A \sim \mu(\cdot \mid s)$, and let

$$F_s(u) \coloneqq \mathbb{P}(M_s \le u)$$

denote its cumulative distribution function (CDF). For a user-specified parameter $\kappa \in [0,1]$, we define the population κ-quantile as

$$q_\kappa(s) \;\coloneqq\; \inf\big\{u \in \mathbb{R} :\; F_s(u) \ge \kappa\big\}.$$

Intuitively, $q_\kappa(s)$ represents the value threshold achieved by the top $(1-\kappa)\times 100\%$ of actions in the behavior policy at state s. The *population preference score* is strictly defined as

$$P_\psi(s,a) \;\coloneqq\; m_\psi(s,a) - q_\kappa(s).$$

In practice, since the ground-truth quantile $q_\kappa(s)$ is inaccessible, we approximate it using an empirical quantile derived from i.i.d. samples $a_1,\ldots,a_N \sim \mu(\cdot \mid s)$. Let $\widehat{q}_{\kappa,N}(s)$ denote the empirical κ-quantile of the set $\{m_\psi(s,a_i)\}_{i=1}^N$. The *empirical preference score* is given by

$$\widehat{P}_N(s,a) \;\coloneqq\; m_\psi(s,a) - \widehat{q}_{\kappa,N}(s). \tag{8}$$

Here, κ serves as a hyperparameter governing the conservatism of the scoring mechanism. When $\kappa \to 1$, the baseline $q_\kappa(s)$ increases, thereby penalizing the action value and imposing a stringent constraint that rejects actions unless they outperform the best responses under the behavior policy. Conversely, as $\kappa \to 0$, the preference score becomes more optimistic, rewarding actions with high potential returns relative to the behavior distribution.

Table 1: Normalized average scores on D4RL tasks, averaged over the final 10 evaluations across 4 seeds. The highest mean scores across all preference-based methods are **bolded**.

	Learning with task rewards		Learning with preference only				
Task	**IQL**	**DQL**	**LSTM**	**PT**	**DPPO**	**PSAC w/o ps**	**PSAC(Ours)**
halfcheetah-mr	$44.3_{\pm 0.7}$	$47.8_{\pm 0.3}$	$42.9_{\pm 1.8}$	$42.3_{\pm 0.5}$	$40.8_{\pm 0.4}$	$41.1_{\pm 0.3}$	**43.3**$_{\pm 0.4}$
halfcheetah-me	$85.2_{\pm 7.4}$	$96.8_{\pm 0.3}$	$77.3_{\pm 4.4}$	$83.6_{\pm 3.8}$	$92.6_{\pm 0.7}$	$91.8_{\pm 0.5}$	**93.0**$_{\pm 1.4}$
hopper-mr	$100.5_{\pm 1.4}$	$101.3_{\pm 0.6}$	$57.9_{\pm 40.6}$	$59.7_{\pm 25.8}$	$73.2_{\pm 4.7}$	$83.4_{\pm 2.8}$	**88.3**$_{\pm 4.7}$
hopper-me	$84.1_{\pm 24.1}$	$111.1_{\pm 1.3}$	$38.6_{\pm 35.6}$	$67.8_{\pm 32.3}$	$107.2_{\pm 5.2}$	$104.9_{\pm 3.2}$	**110.3**$_{\pm 0.4}$
walker2d-mr	$74.8_{\pm 3.4}$	$95.5_{\pm 1.5}$	**77.0**$_{\pm 3.0}$	$43.3_{\pm 39.8}$	$50.9_{\pm 5.1}$	$74.2_{\pm 1.4}$	$76.4_{\pm 0.9}$
walker2d-me	$107.5_{\pm 4.4}$	$110.1_{\pm 0.3}$	**110.4**$_{\pm 0.9}$	$109.8_{\pm 0.4}$	$108.6_{\pm 0.1}$	$88.9_{\pm 1.1}$	$108.5_{\pm 0.5}$
Gym Avg	$82.7_{\pm 6.9}$	$93.8_{\pm 0.7}$	$67.4_{\pm 14.4}$	$67.8_{\pm 17.1}$	$78.9_{\pm 2.7}$	$80.7_{\pm 1.6}$	**86.6**$_{\pm 1.4}$
medium-play	$65.8_{\pm 11.7}$	$76.6_{\pm 10.8}$	$62.9_{\pm 6.9}$	$70.1_{\pm 3.8}$	$5.5_{\pm 3.2}$	$76.5_{\pm 5.4}$	**85.9**$_{\pm 2.4}$
medium-diverse	$73.8_{\pm 5.5}$	$78.6_{\pm 10.3}$	$20.1_{\pm 17.1}$	$65.3_{\pm 3.6}$	$3.6_{\pm 3.3}$	$78.1_{\pm 2.3}$	**84.6**$_{\pm 2.5}$
large-play	$42.0_{\pm 4.5}$	$46.4_{\pm 8.3}$	$14.1_{\pm 3.6}$	$42.4_{\pm 10.0}$	$0.0_{\pm 0.0}$	$54.6_{\pm 5.3}$	**86.8**$_{\pm 3.0}$
large-diverse	$30.3_{\pm 3.6}$	$56.6_{\pm 7.6}$	$0.0_{\pm 0.0}$	$19.6_{\pm 3.7}$	$0.0_{\pm 0.0}$	$29.5_{\pm 6.0}$	**60.4**$_{\pm 2.1}$
AntMaze Avg	$53.0_{\pm 6.3}$	$64.6_{\pm 9.3}$	$24.3_{\pm 6.9}$	$49.4_{\pm 5.3}$	$2.3_{\pm 1.6}$	$59.7_{\pm 4.8}$	**79.4**$_{\pm 2.5}$
pen-human	$53.8_{\pm 36.9}$	$72.8_{\pm 9.6}$	$73.8_{\pm 5.6}$	$53.0_{\pm 31.7}$	$76.3_{\pm 14.4}$	$77.8_{\pm 4.4}$	**80.0**$_{\pm 2.4}$
pen-cloned	$51.3_{\pm 37.1}$	$57.3_{\pm 11.9}$	$60.9_{\pm 5.7}$	$42.9_{\pm 24.4}$	$75.1_{\pm 7.7}$	$70.3_{\pm 2.4}$	**77.6**$_{\pm 2.1}$
hammer-human	$1.4_{\pm 0.6}$	$2.9_{\pm 0.9}$	$1.4_{\pm 0.3}$	$3.0_{\pm 0.6}$	$0.7_{\pm 0.4}$	$3.1_{\pm 0.7}$	**5.1**$_{\pm 0.7}$
hammer-cloned	$1.7_{\pm 1.0}$	$1.5_{\pm 0.4}$	$1.3_{\pm 0.4}$	**2.7**$_{\pm 0.8}$	$2.1_{\pm 0.2}$	$1.5_{\pm 0.7}$	$1.4_{\pm 0.5}$
Adroit Avg	$27.1_{\pm 18.9}$	$33.6_{\pm 5.7}$	$34.4_{\pm 3.0}$	$25.4_{\pm 14.4}$	$38.6_{\pm 5.7}$	$38.2_{\pm 2.1}$	**41.0**$_{\pm 1.4}$
kitchen-complete	$63.8_{\pm 4.4}$	$84.0_{\pm 7.4}$	$48.4_{\pm 8.9}$	$52.3_{\pm 2.4}$	$63.3_{\pm 2.3}$	$78.1_{\pm 4.5}$	**78.4**$_{\pm 3.7}$
kitchen-partial	$50.6_{\pm 6.2}$	$60.5_{\pm 6.9}$	$35.7_{\pm 2.2}$	$40.2_{\pm 12.3}$	$49.4_{\pm 5.7}$	$55.9_{\pm 8.3}$	**70.5**$_{\pm 6.4}$
kitchen-mixed	$58.8_{\pm 6.5}$	$62.6_{\pm 5.1}$	$39.2_{\pm 8.4}$	$48.0_{\pm 11.9}$	$52.5_{\pm 3.1}$	$48.3_{\pm 4.3}$	**57.5**$_{\pm 1.7}$
Kitchen Avg	$57.7_{\pm 5.7}$	$69.0_{\pm 6.5}$	$41.1_{\pm 6.5}$	$46.8_{\pm 8.9}$	$55.1_{\pm 3.7}$	$60.8_{\pm 5.7}$	**68.8**$_{\pm 3.9}$

4 Preference-Score Actor-Critic

Building on the previously described components, we now introduce our algorithm: **P**reference-**S**core **A**ctor-**C**ritic (PSAC).

Diffusion Policy. We parameterize our policy via the reverse process of a conditional diffusion model (Wang et al., 2022):

$$p_\theta(\boldsymbol{a}^{\mathbf{0:T}} \mid \boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}^{\boldsymbol{T}}; \mathbf{0}, \boldsymbol{I}) \prod_{i=1}^{T} p_\theta(\boldsymbol{a}^{\boldsymbol{i-1}} \mid \boldsymbol{a}^{\boldsymbol{i}}, \boldsymbol{s}),$$

where $\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s})$ is the marginal distribution of the terminal sample $\boldsymbol{a}^{\mathbf{0}}$, which serves as the action for RL execution. During the training phase, we sample state-action pairs $(\boldsymbol{s}, \boldsymbol{a})$ from the offline dataset $\mathcal{D}_p$ and generate perturbed samples $\boldsymbol{a}^{\boldsymbol{i}} = \sqrt{\bar{\alpha}_i}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}$ (as per Eq. (19)), with noise schedule parameters defined as $\alpha_i = 1 - \beta_i$, $\bar{\alpha}_i = \prod_{j=1}^{i} \alpha_j$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Adopting the DDPM framework (Ho et al., 2020), we optimize a noise prediction network $\boldsymbol{\epsilon}_\theta(\boldsymbol{a}^{\boldsymbol{i}}, \boldsymbol{s}, i)$ to approximate the injected noise, thereby defining the reverse transition kernel $p_\theta(\boldsymbol{a}^{\boldsymbol{i-1}} \mid \boldsymbol{a}^{\boldsymbol{i}}, \boldsymbol{s})$. The training objective is formulated as:

$$\mathcal{L}_{\mathrm{BC}}(\theta) = \mathbb{E}_{i\sim\mathcal{U}, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}), (\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}_p} \Big[\big\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_i}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}, \boldsymbol{s}, i) \big\|^2 \Big], \tag{9}$$

where $\mathcal{U}$ denotes the uniform distribution over discrete time steps $\{1, \ldots, T\}$. This objective $\mathcal{L}_{\mathrm{BC}}(\theta)$ functions as a behavior cloning (BC) loss; minimizing it ensures that the diffusion model effectively captures the underlying behavior policy μ. During inference, the action $\boldsymbol{a}^{\mathbf{0}}$ is reconstructed by sampling $\boldsymbol{a}^{\boldsymbol{T}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and iteratively applying the learned reverse dynamics.

Preference Score. In our practical implementation, we instantiate the reward model using the non-Markovian architecture described in Eq. (3). This design enables the model to capture complex temporal dependencies and perform explicit credit assignment via importance weighting, optimized

Algorithm 1 Preference-Score Actor-Critic

Initialize all parameters
Sample batched preference data from $\mathcal{D}_p$
Train a reward function r_ϕ from Eq. (2)
Train a value function V_ψ from Eq. (5)
Train a transition model P from Eq. (10)
for each iteration **do**
  Obtain $\widehat{P}_N(\boldsymbol{s'}, \boldsymbol{a'})$ according to Eq. (8) *// Preference Score Calculation*
  Update $\mathcal{L}_{\text{CRITIC}}(\varphi)$ according to Eq. (11) *// Critic Update*
  Update $\mathcal{L}_{\text{ACTOR}}(\theta)$ according to Eq. (12) *// Actor Update*
  Soft update parameter φ and θ *// Target Network Update*
end for

by minimizing the preference-based cross-entropy loss in Eq. (2). Simultaneously, we also optimize the value function directly from the preference dataset. This process is governed by the probability parameterization in Eq. (4) and minimized via the classification objective in Eq. (5).

To compute the empirical preference score in Eq. (8), we employ a learned transition model to approximate the expectation over system dynamics as defined in Eq. (7). We parameterize the transition dynamics using a neural network and learn a deterministic transition model, which we find sufficiently accurate and computationally efficient in practice. The model is trained by minimizing the following mean squared error (MSE) loss:

$$\mathcal{L}_{\text{P}}(\varsigma) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})\sim\mathcal{D}_p}\left[\|P_\varsigma(\boldsymbol{s},\boldsymbol{a}) - \boldsymbol{s'}\|^2\right]. \tag{10}$$

Therefore, the empirical preference score function $\widehat{P}_N$ is then obtained as defined in Eq. (8). All components are trained jointly using only offline data and then kept fixed during subsequent actor–critic updates, making this stage computationally inexpensive.

Actor-Critic. Following the preference score defined in Eq. (8), we design the loss for learning Q-function (critic) as:

$$\begin{aligned}\mathcal{L}_{\text{CRITIC}}(\varphi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})\sim\mathcal{D}_p,\boldsymbol{a'}\sim\pi_\theta(\cdot|\boldsymbol{s'})}[(r_\phi(\boldsymbol{s},\boldsymbol{a}) \\ + \gamma\big(Q_\varphi(\boldsymbol{s'},\boldsymbol{a'}) + \lambda\widehat{P}_N(\boldsymbol{s'},\boldsymbol{a'})\big) - Q_\varphi(\boldsymbol{s},\boldsymbol{a}))^2].\end{aligned} \tag{11}$$

Here, the preference score function $\widehat{P}_N(\boldsymbol{a} \mid \boldsymbol{s})$ acts as an auxiliary correction term. Since the learned reward r_ϕ acts merely as an imperfect proxy rather than ground truth, $\widehat{P}_N$ serves as an independent evaluative reference derived directly from preference rankings. Integrating this term allows us to rectify the Q-value learning, effectively compensating for potential biases or errors inherent in the reward model.

To improve the policy, we incorporate Q-function guidance into the behavior cloning objective, encouraging the model to sample actions with greater estimated values. The resulting policy (actor) objective combines policy regularization and policy improvement:

$$\mathcal{L}_{\text{ACTOR}}(\theta) = \mathcal{L}_{\text{BC}}(\theta) - \alpha\mathbb{E}_{\boldsymbol{s}\sim\mathcal{D}_p,\boldsymbol{a}\sim\pi_\theta}\left[Q_\varphi(\boldsymbol{s},\boldsymbol{a})\right]. \tag{12}$$

We summarize our implementation in Algorithm 1, which is a preference score augmented actor-critic algorithm.

5 Experiments

We evaluate PSAC on 17 D4RL tasks, ablate the preference-score augmentation and value-learning objective, vary the preference-query budget, and test high-level humanoid navigation in simulation.

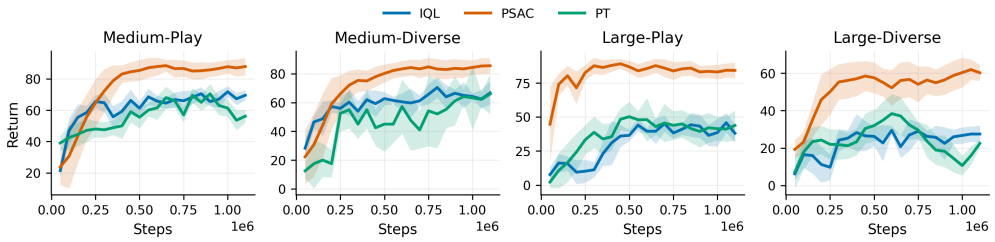


Figure 2: **Learning curves on AntMaze benchmarks.** We compare PSAC against IQL and PT. The solid lines represent the mean return averaged over 4 random seeds, while the shaded regions indicate the standard deviation.

Table 2: **Ablation on value learning paradigms.** We compare our 'Direct' approach against the 'Indirect' expectile regression baseline. Results show mean ± standard deviation over 4 random seeds on Adroit and Kitchen tasks.

Task	Indirect	Direct
franka-kitchen-complete	$\mathbf{81.7 \pm 5.3}$	78.4 ± 3.7
franka-kitchen-partial	54.2 ± 4.9	$\mathbf{70.5 \pm 6.4}$
franka-kitchen-mixed	49.5 ± 4.4	$\mathbf{57.5 \pm 1.7}$
adroit-pen-human	78.2 ± 3.4	$\mathbf{80.0 \pm 2.4}$
adroit-pen-clone	76.7 ± 2.8	$\mathbf{77.6 \pm 2.1}$
adroit-hammer-human	3.8 ± 0.8	$\mathbf{5.1 \pm 0.7}$
adroit-hammer-clone	$\mathbf{1.7 \pm 0.5}$	1.4 ± 0.5
Average	49.4 ± 3.2	$\mathbf{52.9 \pm 2.5}$

5.1 Evaluation on D4RL Benchmarks

Dataset. We evaluate our method on four distinct domains from the D4RL benchmark. **Gym.** These tasks are widely adopted and straightforward due to dense reward signals. **AntMaze.** In contrast, this domain presents challenging sparse-reward scenarios, requiring the composition of suboptimal trajectories. **Adroit.** Collected from human demonstrations, these tasks involve narrow state-action regions and demand strong regularization. **Kitchen.** This environment poses a multi-task control problem requiring sequential completion of sub-tasks, emphasizing long-term planning.

Adhering to the standard offline PbRL protocol, our setup comprises a large unlabeled dataset $\mathcal{D}$ sourced from D4RL (Fu et al., 2020) and a sparse set of preference-labeled segments $\mathcal{D}_p$. For preference supervision, we directly leverage the human-labeled datasets released by PT (Kim et al., 2023) and DPPO (An et al., 2023). This strategy eliminates discrepancies in label generation, ensuring rigorous benchmarking under identical supervision conditions. Further details on task specifications and dataset statistics are provided in Appendix A.

Baseline. We benchmark our approach against a diverse set of strong baseline methods. For non-Markovian reward modeling, we evaluate against LSTM Early et al. (2022) and PT (Kim et al., 2023). For methods of direct policy optimization without reward learning, we compare DPPO An et al. (2023). Additionally, we include IQL and DQL trained on ground-truth task rewards to serve as oracle references. We report the performance of baseline methods either from their original paper or from our own reproduction, enabling a fair comparison.

Table 1 summarizes the results across four D4RL domains. PSAC has the highest mean score among the compared preference-based methods on 14 of 17 tasks. Breaking down performance by domain, PSAC achieves substantial margins over prior preference-based methods: approximately **10%** in Gym, **6%** in Adroit, **24%** in Kitchen, and a pronounced **60%** improvement in the challenging AntMaze tasks. Compared with its ablated counterpart (w/o ps), PSAC improves the reported domain-average score in each of the four domains, supporting the preference-score augmentation. Given the remarkable gains in AntMaze, we plot the learning curves in Figure 2 to explicitly demonstrate

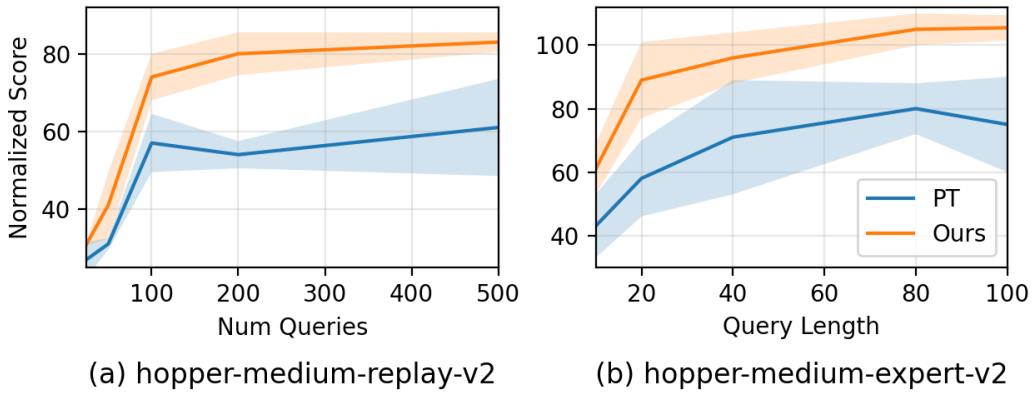


Figure 3: **Data-efficiency analysis on Hopper tasks.** We compare PSAC against PT by varying (a) the number of preference queries on `hopper-medium-replay-v2` and (b) the query segment length on `hopper-medium-expert-v2`. The solid lines represent the mean normalized score averaged over 4 random seeds, and the shaded regions indicate the standard deviation.

our method's superior performance against the second-best preference-based method (PT) and the ground-truth reward baseline (IQL).

PSAC also exceeds the domain-average scores of IQL and DQL trained with task rewards on AntMaze and Adroit, while its Kitchen average is close to DQL. These comparisons show that preference supervision can be competitive with engineered task rewards on the evaluated domains; they do not establish that preferences are universally more informative.

5.2 Efficacy of Direct Value Learning

In Subsection 3.2, we proposed a method to directly learn a value function from the preference dataset. To validate the reliability of this approach, we benchmark it against the conventional expectile regression baseline (denoted as 'Indirect' in Table 2), as defined in Eq. (6).

Direct value learning improves the reported average from 49.4 ± 3.2 to 52.9 ± 2.5 (Table 2). It achieves higher means on five of seven tasks, with lower means on kitchen-complete and hammer-cloned. Thus, its average benefit is accompanied by task-dependent trade-offs.

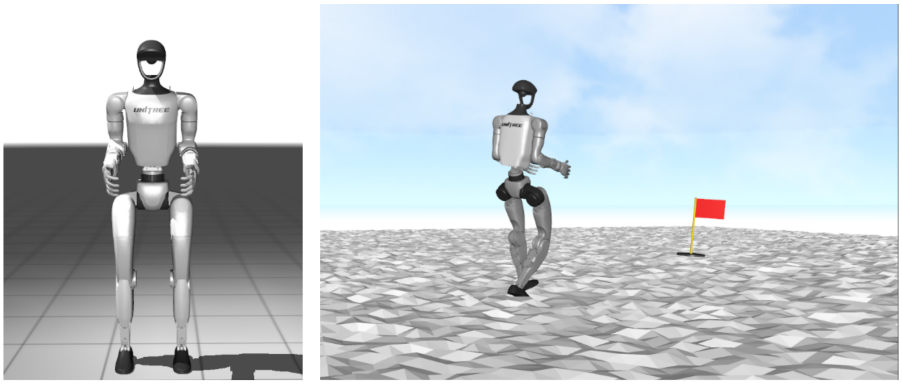
5.3 Data-Efficiency Analysis

Preference annotation is costly. We therefore compare PSAC and PT under varying query budgets and segment lengths on Hopper, testing whether the two uses of preference data improve data efficiency.

Figure 3 varies the number of comparisons on hopper-medium-replay and the segment length on hopper-medium-expert. PSAC performs better under the evaluated budgets, including fewer comparisons and shorter segments. The results are consistent with the benefit of learning both reward and preference-score signals from the same feedback.

5.4 Scalability to Humanoid Robotics

To evaluate PSAC in a realistic setting, we conduct experiments on the G1 `GoalReach-RoughTerrain` task (see Figure 4) using the Unitree G1 robot within MuJoCo Playground (Zakka et al., 2025). The robot must navigate uneven terrain toward visual goals. We collect teleoperated demonstrations to extract trajectory segments of length 100 and obtain 676 human preference queries based on task completion, orientation, and gait stability.



Unitree G1 Robot G1-GoalReach-RoughTerrain

Figure 4: **Visualizations of the humanoid experimental setup.** Left: The `Unitree G1` humanoid robot model used in the simulations. Right: The `G1-GoalReach-RoughTerrain` task environment in MuJoCo, depicting the robot navigating across uneven terrain toward a goal marked by a red flag.

Table 3: **Performance evaluation over 140 trials.** We compare PSAC with the BC baseline across five metrics: Success rate (Succ↑), fraction of fall episodes (Fall↓), average cosine heading alignment (Heading↑), task completion time (Duration↓), and squared-torque cost (Cost↓, calculated as $\int \|\tau_t\|_2^2 dt$).

Method	Succ%↑	Fall%↓	Heading↑	Duration↓	Cost↓
BC	61.0	30.5	0.636	21.99	7846.87
PSAC	**72.3**	**20.6**	**0.701**	**17.40**	**7653.11**

Over 140 evaluation trials, PSAC raises the reported success rate from 61.0% to 72.3% and reduces the fall rate from 30.5% to 20.6% relative to BC (Table 3). It also improves heading alignment, duration, and the reported squared-torque cost. PSAC controls high-level velocity commands; the pre-trained low-level controller uses shaped locomotion rewards (Appendix B.5). These simulation results support preference-based navigation without establishing performance on physical hardware.

6 Conclusion

PSAC augments offline actor–critic learning with a preference score derived directly from comparisons. It attains the highest mean score among the compared preference-based methods on 14 of 17 D4RL tasks and improves preference-data efficiency on the evaluated Hopper tasks. With 676 labeled pairs, it also improves high-level rough-terrain navigation over BC in a Unitree G1 simulation.

AI Use Statement

An AI assistant helped edit the language and structure, format the manuscript, and check consistency between the text, equations, and reported results. It also assisted with interpreting reported comparisons and clarifying mathematical exposition. Experimental results were retained from the source manuscript.

References

Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In *Joint European conference on machine learning and knowledge*

discovery in databases, pp. 116–131. Springer, 2012.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Gaon An, Junhyeok Lee, Xingdong Zuo, Norio Kosaka, Kyung-Min Kim, and Hyun Oh Song. Direct preference-based policy optimization without reward modeling. *Advances in Neural Information Processing Systems*, 36:70247–70266, 2023.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

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

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Joseph Early, Tom Bewley, Christine Evers, and Sarvapali Ramchurn. Non-markovian reward modelling from trajectory labels via interpretable multiple instance learning. *Advances in Neural Information Processing Systems*, 35:27652–27663, 2022.

Aleksandra Faust, Anthony Francis, and Dar Mehta. Evolving rewards to automate reinforcement learning. *arXiv preprint arXiv:1905.07628*, 2019.

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Joey Hejna and Dorsa Sadigh. Inverse preference learning: Preference-based rl without a reward function. *Advances in Neural Information Processing Systems*, 36:18806–18827, 2023.

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

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Daniel Kahneman. Evaluation by moments: Past and future. *Choices, values, and frames*, pp. 693–708, 2000.

Changyeon Kim, Jongjin Park, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Preference transformer: Modeling human preferences using transformers for rl. *arXiv preprint arXiv:2303.00957*, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

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

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pp. 45–73. Springer, 2012.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. Active preference-based learning of reward functions. 2017.

Connor Schenck and Dieter Fox. Visual closed-loop control for pouring liquids. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2629–2636. IEEE, 2017.

Daniel Shin, Anca D Dragan, and Daniel S Brown. Benchmarks and algorithms for offline preference-based reward learning. *arXiv preprint arXiv:2301.01392*, 2023.

David Silver, Richard S Sutton, and Martin Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, pp. 968–975, 2008.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

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

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. *Advances in neural information processing systems*, 25, 2012.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020a.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020b.

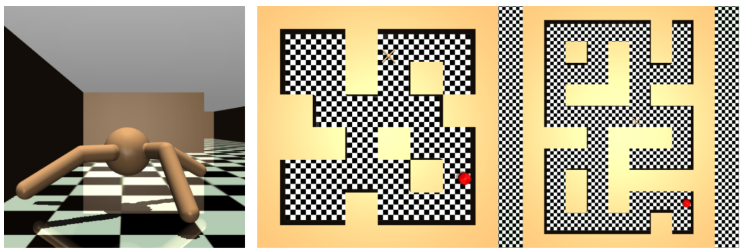
Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

A DATASETS AND TASKS

Our evaluation covers D4RL domains (`Gym`, `AntMaze`, `Adroit`, `Franka Kitchen`) and an in-house dataset, G1 `GoalReach-RoughTerrain`.

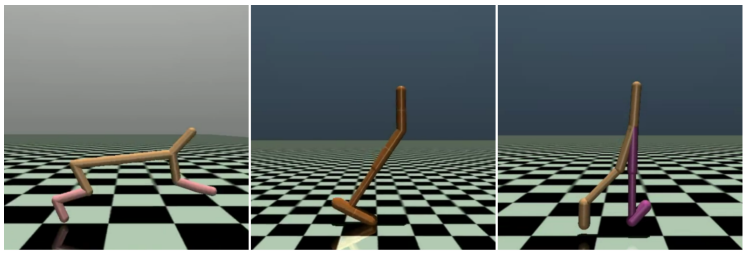
AntMaze is a sparse-reward navigation benchmark where an ant robot must traverse maze layouts of varying complexity to reach a goal location. The main challenge is long-horizon planning with delayed credit assignment under sparse success signals and many dead ends. We evaluate on `medium-play`, `medium-diverse`, `large-play`, and `large-diverse`. For preference collection, we use 50-step trajectory segments and collect 1000 queries per dataset.



Ant robot Medium maze Large maze

Figure 5: **AntMaze.** Illustration of the ant robot (left), a medium maze layout (middle), and a large maze layout (right).

Gym Locomotion comprises standard continuous-control tasks where agents must move forward while maintaining balance and stability. These tasks provide dense environment rewards that correlate with forward velocity, while preference feedback can further emphasize stability and smoothness. We use the `medium-replay` and `medium-expert` datasets, covering a wide range of trajectory quality. For preference collection, we use 50-step segments; we collect 500 queries for `medium-replay` and 100 queries for `medium-expert` per dataset.



Halfcheetah Hopper Walker2D

Figure 6: **Gym Locomotion.** Screenshots of the three locomotion agents: HalfCheetah (left), Hopper (middle), and Walker2d (right).

Adroit Hands evaluates dexterous, contact-rich manipulation with a 24-DoF robotic hand, requiring precise coordination in a high-dimensional continuous action space. We consider `Pen` and `Hammer` due to their strong contact dynamics and control sensitivity. For preference collection, we use 50-step trajectory segments and collect 100 queries per dataset.

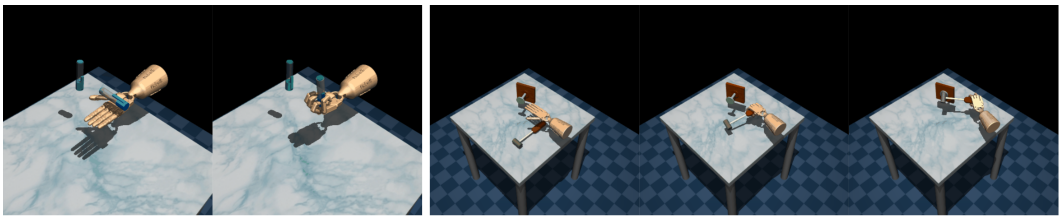


Figure 7: **Adroit Hands.** Example rollouts for Pen (left, middle) and Hammer (right) showing successful task completion.

Franka Kitchen is a long-horizon 9-DoF manipulation benchmark that requires completing multiple household sub-tasks within a single episode, highlighting compositionality and goal ordering (e.g., open microwave, move kettle, flip light switch, slide cabinet door). We evaluate on `complete`, `partial`, and `mixed`, which differ in task completion and demonstration diversity. For preference collection, we use 50-step trajectory segments and collect 100 queries per dataset.

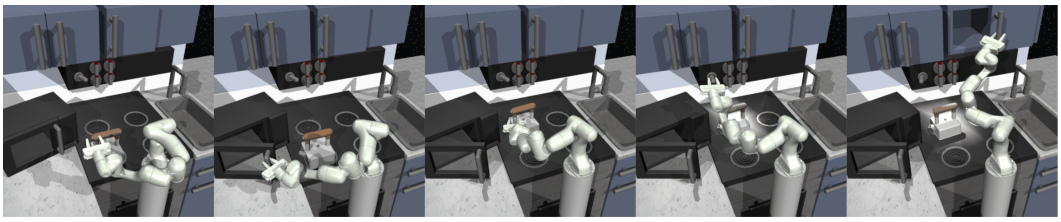


Figure 8: **Franka Kitchen.** A sequence of snapshots from initial state to the completion of four ordered sub-tasks: open microwave, move kettle, flip light switch, and slide cabinet door.

G1-GoalReach-RoughTerrain is an in-house dataset based on a Unitree G1 humanoid robot model controlled via a Microsoft Xbox 360 joystick. The environment is implemented in MuJoCo (MuJoCo Playground), where the robot navigates uneven terrain toward visually indicated goal locations. Human preference labels are determined by three criteria: *task completion*, *orientation consistency*, and *gait stability*. We use trajectory segments of length 100 and collect up to 676 preference queries.

B Experiment Details

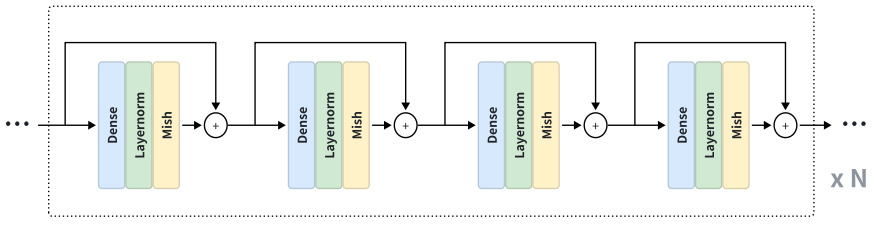


Figure 9: Residual block architecture used in critic networks.

B.1 Network Architecture

Our method jointly optimizes three network modules during training: a diffusion-based actor, a Q-function critic, and a value function V. The actor and critic architectures follow the same design as those used in our main experiments, and are shared across different algorithmic variants. This section

provides architectural details for completeness, while excluding auxiliary networks used solely for preference-score estimation.

Diffusion Actor. The actor is instantiated as a denoising diffusion probabilistic model (DDPM) with a variance-preserving (VP) noise schedule. The noise predictor is implemented as a five-layer multilayer perceptron (MLP) with Mish activations. We fix the number of denoising steps to 10 across all tasks, which provides sufficient expressive power while maintaining computational efficiency. The diffusion actor outputs continuous actions conditioned on state inputs, and is trained jointly with the critic during policy optimization.

Critic Networks. Both the Q-function $Q(s, a)$ and the value function $V(s)$ are parameterized by residual networks. Each critic is composed of a stack of L=16 residual blocks. Given an input feature vector $h^{(0)}$ (state for V, and state–action for Q), the l-th residual block is defined as

$$h^{(l+1)} = h^{(l)} + f^{(l)}\left(h^{(l)}\right), \tag{13}$$

where $f^{(l)}(\cdot)$ denotes a four-layer multilayer perceptron with a non-linear activation. The output of the final block is mapped to a scalar value estimate through a linear projection. An illustration of the residual block is provided in Figure 9.

Preference Transformer. We parameterize the preference model as a decoder-only causal Transformer in the GPT-2 style. Given a trajectory segment τ of length T, we construct a token sequence $\{z_t\}_{t=1}^{T}$ using per-timestep embeddings of the segment inputs. The Transformer applies stacked causal self-attention blocks to produce contextual representations $\{u_t\}_{t=1}^{T}$.

To obtain a segment-level score, we apply an attention-based temporal pooling over $\{u_t\}$. Concretely, we compute a scalar logit for each timestep and normalize it into a weight:

$$e_t = w^\top u_t, \qquad \alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^{T} \exp(e_j)}, \qquad \bar{u} = \sum_{t=1}^{T} \alpha_t u_t, \tag{14}$$

where w is a learnable vector and α_t are content-dependent weights. The segment score is then given by a linear head

$$s(\tau) = v^\top \bar{u} + b. \tag{15}$$

For a pair of segments $(\tau^{(1)}, \tau^{(2)})$, the preference probability is modeled with a Bradley–Terry likelihood:

$$P\left(\tau^{(1)} \succ \tau^{(2)}\right) = \frac{\exp\left(s(\tau^{(1)})\right)}{\exp\left(s(\tau^{(1)})\right) + \exp\left(s(\tau^{(2)})\right)}. \tag{16}$$

Preference Value Function. We parameterize the preference value function $V_\psi(s)$ as an MLP. This state-only model is learned from single-step preference comparisons and is fitted solely on state observations collected from trajectories. Given a preference-labeled pair of states $(s^{(1)}, s^{(2)})$, the Bradley–Terry preference likelihood is defined as

$$P_\psi\left(s^{(1)} \succ s^{(2)}\right) = \frac{\exp\left(V_\psi(s^{(1)})\right)}{\exp\left(V_\psi(s^{(1)})\right) + \exp\left(V_\psi(s^{(2)})\right)}. \tag{17}$$

The preference value function is optimized by minimizing a binary cross-entropy loss constructed from this likelihood.

B.2 Hyperparameters

Tables 4 and 5 summarize the hyperparameter settings used in our experiments. Table 4 reports the shared configurations for the reward model (Preference Transformer), the preference value function, and the diffusion actor–critic used for policy learning. Table 5 reports the task-dependent settings, including the max-Q backup configuration, the Q-guidance weight α, and the preference-query protocol.

Table 4: Summary of hyperparameters for preference models and policy learning.

Module	Hyperparameter	Value
Reward Model (PT)	Backbone	Causal Transformer
	Input	State and action sequences
	# of Transformer layers	1
	# of Attention heads	4
	Embedding dimension	256
	Batch size	256
	Dropout (embedding / attention / residual)	0.1
	Learning rate	1×10^{-4}
	Optimizer	AdamW (Loshchilov & Hutter, 2017)
	AdamW momentum	$\beta_1 = 0.9,\ \beta_2 = 0.99$
	Weight decay	1×10^{-4}
	Warmup steps	500
	Total gradient steps	10K
	Query length	task-dependent L
	Query budget	task-dependent B
Preference Value Function	Backbone	Residual MLP
	Input	Single-step state
	# of Residual blocks	2
	# of Residual block layers	4
	Batch size	256
	Learning rate	1×10^{-4}
	Optimizer	AdamW (Loshchilov & Hutter, 2017)
	Total gradient steps	10K
Diffusion Policy	Actor type	Diffusion policy (DDPM)
	Noise schedule	Variance-preserving (VP)
	# of Denoising steps	10
	Noise predictor	5-layer MLP + Mish
	Critic backbone	Residual MLP
	# of Residual blocks	4
	# of Residual block layers	4
	Actor optimizer	AdamW (Loshchilov & Hutter, 2017)
	Critic optimizer	AdamW (Loshchilov & Hutter, 2017)
	Policy learning rate grid	$\{10^{-3}, 3 \times 10^{-4}, 3 \times 10^{-5}\}$
	Preference score weight λ	1.0
	Q-guidance weight α	task-dependent
	Max-Q backup	task-dependent

For the preference models (Table 4), the reward model is implemented as a causal Transformer over state–action sequences, with one Transformer layer, four attention heads, and a 256-dimensional embedding. It is trained using AdamW with a fixed batch size, dropout, warmup schedule, and total training steps. The preference value function is a residual MLP on single-step states, trained with the same optimizer family and training budget.

For policy learning (Table 4), the actor is a DDPM with a variance-preserving noise schedule and 10 denoising steps, using a five-layer MLP with Mish activations as the noise predictor. The critic uses a residual MLP backbone, and both actor and critic are optimized with AdamW; the policy learning rate is selected from the grid reported in the table.

Task-specific settings are listed in Table 5. For each dataset, we report whether max-Q backups are enabled and the number of backup repetitions n. We also report the Q-guidance weight α, together with the preference-query segment length L and the query budget B used to train the preference models.

Table 5: Task-specific hyperparameters. We report whether max-Q backups are applied, the number of backup times n, the diffusion policy's guidance weight α, and the preference query protocol (segment length L and query budget B). Query budgets for Gym are reported *per dataset* (e.g., HalfCheetah/Hopper/Walker2d each uses the same B under the same dataset type).

Task	Max Q Backup	n	α	L	B
halfcheetah-medium-replay	True	5	1.0	50	500
halfcheetah-medium-expert	True	10	1.0	50	100
hopper-medium-replay	True	5	1.0	50	500
hopper-medium-expert	False	1	1.0	50	100
walker2d-medium-replay	False	1	1.0	50	500
walker2d-medium-expert	False	1	1.0	50	100
antmaze-medium-play	True	10	2.0	50	1000
antmaze-medium-diverse	True	10	3.0	50	1000
antmaze-large-play	True	10	4.5	50	1000
antmaze-large-diverse	True	10	3.5	50	1000
pen-human	True	3	0.15	50	100
pen-cloned	True	3	0.10	50	100
hammer-human	True	3	0.15	50	100
hammer-cloned	True	3	0.10	50	100
kitchen-complete	False	1	0.005	50	100
kitchen-partial	False	1	0.005	50	100
kitchen-mixed	False	1	0.005	50	100

B.3 PSAC WITH INDIRECT PREFERENCE VALUE FUNCTION

We use an indirect PSAC variant (Alg. 2) in Sec. 5.2 for the experiments that require joint optimization of the preference value estimator and the policy on offline data. In this variant, the indirect preference value function is updated within the main training loop and co-trained with the diffusion actor–critic, rather than being trained in a separate stage. This contrasts with Direct PSAC in the main paper, where the preference value function is pre-trained before actor–critic iterations and then used to guide subsequent updates.

Algorithm 2 Indirect Preference-Score Actor-Critic

Initialize all parameters
Sample batched preference data from $\mathcal{D}_p$
Train a reward function r_ϕ from Eq. (2)
Train a transition model P from Eq. (10)
for each iteration **do**
    Update $\mathcal{L}_V(\psi)$ according to Eq. (6)
    Obtain $\widehat{P}_N(\boldsymbol{s'}, \boldsymbol{a'})$ according to Eq. (8)
    Update $\mathcal{L}_{\text{CRITIC}}(\varphi)$ according to Eq. (11)
    Update $\mathcal{L}_{\text{ACTOR}}(\theta)$ according to Eq. (12)
    Soft update parameter φ and θ
end for

B.4 BASELINES

We benchmark our approach against a diverse set of strong PbRL baselines, comprising both non-Markovian reward modeling techniques and direct policy learning methods.

LSTM. A non-Markovian reward method proposed by Early et al. (2022) that employs an LSTM to capture sequential information during reward learning.

PT. A non-Markovian approach that utilizes Transformer architectures to model temporal dependencies, facilitating more accurate credit assignment.

Table 6: **Joystick-G1 low-level reward terms.** The per-step locomotion reward is a weighted sum of terms from the official implementation: $r_t = \Delta t \sum_k s_k\, t_k$, where Δt is the control time step (here $\Delta t = 0.02$). Commands are $\mathbf{c} = (c_x, c_y, c_\omega)$ and local pelvis velocity is $\mathbf{v}^{\text{loc}}$. We use $\sigma = 0.25$ (tracking kernel), $z_{\max} = 0.15$ (foot-height target), $h^\star = 0.5$ (base-height target), and $F_{\max} = 500$ (contact-force threshold). Negative s_k indicate penalties.

Term t_k	**Definition**	**Scale** s_k
`tracking_lin_vel`	$\exp\Big(-\lVert\mathbf{c}_{xy} - \mathbf{v}^{\text{loc}}_{xy}\rVert^2/\sigma\Big)$	$+1.00$
`tracking_ang_vel`	$\exp\Big(-(c_\omega - \omega^{\text{gyro}}_z)^2/\sigma\Big)$	$+0.75$
`lin_vel_z`	$(v^{\text{glob}}_{\text{torso},z})^2 + (v^{\text{glob}}_{\text{pelvis},z})^2$	0.00
`ang_vel_xy`	$\lVert\boldsymbol{\omega}^{\text{glob}}_{\text{torso},xy}\rVert_2^2$	-0.15
`orientation`	$\lVert\mathbf{g}_{\text{torso}} - [0.073,\ 0,\ 1]\rVert_2^2$	-2.00
`base_height`	$(h - h^\star)^2$	0.00
`torques`	$\sum_i \lvert\tau_i\rvert$	0.00
`action_rate`	$\sum_i (a_{t,i} - a_{t-1,i})^2$	0.00
`energy`	$\sum_i \lvert\dot{q}_i\rvert\,\lvert\tau_i\rvert$	0.00
`dof_acc`	$\sum_i \ddot{q}_i^2$	0.00
`feet_clearance`	$\sum_f \lvert z_f - z_{\max}\rvert\ \sqrt{\lVert\mathbf{v}_{f,xy}\rVert_2}$	0.00
`feet_height`	$\sum_f \left(\frac{z_f^{\text{peak}}}{z_{\max}} - 1\right)^2 \cdot \mathbb{I}[\text{first_contact}_f]$	0.00
`feet_air_time`	$\sum_f \text{clip}\Big((T_f - 0.2)\mathbb{I}[\text{first_contact}_f],\ 0,\ 0.3\Big)$	$+2.00$
`feet_phase`	$\exp\Big(-\sum_f (z_f - r_z(\phi_f))^2/0.01\Big) \cdot \mathbb{I}[\text{moving}]$	$+1.00$
`feet_slip`	$\lVert\mathbf{v}^{\text{glob}}_{\text{pelvis},xy}\rVert_2 \cdot \sum_f \mathbb{I}[\text{contact}_f]$	-0.25
`alive`	1	0.00
`stand_still`	$\mathbb{I}[\lVert\mathbf{c}\rVert_2 < 0.01] \sum_i \lvert q_i - q_i^{\text{default}}\rvert$	-1.00
`termination`	$\mathbb{I}[\text{done}]$	-100.00
`collision`	$\mathbb{I}[\text{self-collision}]$	-0.10
`contact_force`	$\sum_{f\in\{\text{L,R}\}} \text{clip}\big(\lvert F_{f,z}\rvert - F_{\max},\ 0,\ \infty\big)$	-0.01
`joint_deviation_hip`	$\sum_{j\in\mathcal{J}_{\text{hip}}} \lvert q_j - q_j^{\text{default}}\rvert \cdot w_j(\mathbf{c})$	-0.25
`joint_deviation_knee`	$\sum_{j\in\mathcal{J}_{\text{knee}}} \lvert q_j - q_j^{\text{default}}\rvert$	-0.10
`dof_pos_limits`	$\sum_i \Big[-\text{clip}(q_i - \ell_i,\ -\infty, 0) + \text{clip}(q_i - u_i, 0, \infty)\Big]$	-1.00
`pose`	$\sum_i (q_i - q_i^{\text{default}})^2$	-0.10

DPPO. A direct optimization method (An et al., 2023) that derives optimal policies immediately from preference constraints, bypassing the need to learn a reward model.

B.5 Experiments on Humanoid Robotics

Hierarchical control setup. We evaluate PSAC in a physics-based simulation of the Unitree G1 humanoid under a two-level control hierarchy. Our learned policy operates at a *high level*, outputting navigation commands, while a pre-trained *low-level locomotion controller* executes stable walking on rough terrain. Concretely, the low-level controller tracks a commanded planar velocity and yaw rate, i.e., (v_x, v_y, ω_z), enabling the high-level policy to *remote-control* the humanoid to move in arbitrary directions while maintaining a reasonably upright posture. This hierarchy isolates long-horizon decision making from low-level contact dynamics: the high-level policy only needs to reason about goal-directed motion and heading, whereas the low-level controller handles balance, foot placement, and disturbance rejection. In all experiments, the state and action spaces of PSAC correspond to this high-level interface (commanded (v_x, v_y, ω_z)), and all rollouts are executed in full physics.

Low-level controller training and sim-to-sim transfer. The low-level locomotion controller is trained following the *rough-terrain* recipe in MuJoCo Playground (Zakka et al., 2025), using MJX for massively-parallel training. The controller is conditioned on joystick-style commands (v_x, v_y, ω_z) and is optimized with the *official* shaped locomotion reward used by MuJoCo Playground. This

reward has the standard structure of (i) command tracking (linear-velocity and yaw-rate tracking) and (ii) stability/regularization terms that encourage robust walking and penalize physically implausible behaviors. The latter include posture/orientation stabilization, base-height and velocity-related penalties, action-rate and energy costs, as well as foot-related shaping such as air-time, phase tracking, slip, clearance, and contact-force penalties, together with termination and self-collision signals. Concretely, the per-step reward is computed as

$$r_t \;=\; \Delta t \sum\nolimits_k s_k \, t_k, \tag{18}$$

where $\{t_k\}$ are the individual reward/cost terms and $\{s_k\}$ are the default scaling coefficients (with negative s_k indicating penalties), and Δt is the control time step. For completeness and reproducibility, we list all reward terms, their code-faithful definitions, and the corresponding coefficients in Table 6.

After training in MJX, we perform *scene-to-scene* sim-to-sim transfer by evaluating the same controller in the standard MuJoCo CPU simulator (MuJoCo Playground CPU backend), which we use for all reported rollouts and metrics. This transfer is important for two reasons: (i) it aligns the reported evaluation with the standard MuJoCo pipeline and its diagnostics, and (ii) it reduces the possibility that gains are tied to a specific simulator backend. In our setup, MJX is used exclusively to accelerate locomotion training, while all offline data collection and policy evaluation are conducted in the CPU simulator under identical low-level control code.

Trajectory collection via teleoperation. To build the offline dataset for high-level learning, we collect trajectories by teleoperating the humanoid through the low-level command interface. At the start of each episode, we randomly sample a target position in the plane and generate joystick-style commands (v_x, v_y, ω_z) to drive the robot toward the goal while maintaining a stable, forward-facing walking posture. This procedure yields long-horizon navigation trajectories that are *dynamically feasible* under the low-level controller and naturally include the characteristic constraints of rough-terrain locomotion (e.g., limited turning while stepping, transient slips, and occasional recovery behaviors). Importantly, because the high-level actions correspond to smooth velocity commands rather than low-level torques, the collected dataset provides a realistic substrate for studying preference-driven learning of goal-directed behaviors at the decision-making level.

Preference dataset construction. In addition to trajectories, PSAC requires pairwise preferences. We construct preference queries by slicing fixed-length segments of length $L{=}100$ from the collected trajectories and sampling segment pairs for comparison. To ensure that comparisons remain informative across different phases of an episode (and avoid the degenerate case where both segments are trivially poor or indistinguishable), we use a structured pairing scheme: 25% of pairs are sampled from the *first third* of each trajectory, 25% from the *middle third*, 25% from the *last third*, and the remaining 25% are sampled uniformly at random over the entire trajectory. This mixture injects preference signals about early-stage departure behavior (e.g., promptly orienting and accelerating toward the goal), mid-trajectory progress and stability (e.g., avoiding unnecessary oscillations or detours), and late-stage goal-reaching behavior (e.g., precise approach and stopping near the target), while retaining broad coverage that correlates with global measures such as distance-to-goal. We label 676 preference pairs in total, requiring roughly two hours of human annotation, and use them to train the preference model that provides the preference score/value signals for PSAC.

Baselines and additional observations. We report results for Behavior Cloning (BC) and our preference-based RL method (PSAC) in table 3. We also experimented with learning from a sparse task reward using the same diffusion policy architecture with Q-guidance. Empirically, large guidance weights (e.g., $\lambda{=}1.0$) lead to near-complete failures, consistent with brittle long-horizon credit assignment and distributional shift under sparse feedback in humanoid navigation on rough terrain. As λ decreases, performance gradually approaches BC; however, across the tested range, it remains consistently worse than BC as $\lambda \to 0$. These observations motivate preference-driven training in this setting: preferences provide denser, behavior-level supervision that is more aligned with stable goal-reaching and is less sensitive to the tuning instability that arises when naively amplifying sparse rewards through guidance.

C Related Work

Preference-based Reinforcement Learning. Leveraging human preferences has emerged as a powerful paradigm for guiding agent behavior (Akrour et al., 2012; Ibarz et al., 2018), demonstrating

remarkable success in complex control tasks (Christiano et al., 2017; Lee et al., 2021) and large language model alignment (Stiennon et al., 2020; Ouyang et al., 2022; Rafailov et al., 2023). In the context of offline PbRL, Shin et al. (2023) established the OPRL benchmark to evaluate various baselines, though many rely on computationally intensive disagreement-based query selection. To enhance the expressivity of reward modeling, Early et al. (2022) introduced the LSTM-based architecture as a reward function. Kim et al. (2023) proposed the Preference Transformer (PT), which utilizes Transformer architectures to capture non-Markovian temporal dependencies for more accurate credit assignment. However, recent advancements have begun exploring approaches that eliminate the separate reward learning phase in favor of direct policy optimization. An et al. (2023) introduced DPPO to directly derive optimal policies from preference constraints. Similarly, Hejna & Sadigh (2023) proposed Inverse Preference Learning (IPL), which completely eschews explicit reward modeling by leveraging the bijective relationship between Q-functions and rewards via the inverse soft-Bellman operator. Distinct from these paradigms, we propose Preference-Score Actor-Critic (PSAC), which augments the critic update with a preference score grounded in relative comparisons, providing an auxiliary signal alongside the learned proxy reward.

Offline Reinforcement Learning. Offline RL (Lange et al., 2012; Levine et al., 2020) focuses on learning effective policies solely from static datasets, which enables PbRL to be solved via standard offline algorithms once the reward function is learned. Model-free offline RL methods are primarily divided into pessimistic value-based and regularized policy-based approaches. Pessimistic methods discourage overestimation by penalizing Q-values for out-of-distribution actions, utilizing either uniform penalties like CQL (Kumar et al., 2020) or uncertainty-adjusted penalties as in EDAC (An et al., 2021). Meanwhile, regularized policy-based methods explicitly constrain the policy to the behavior distribution, ranging from hard constraints or action-space restrictions in BEAR (Kumar et al., 2019) and BCQ (Fujimoto et al., 2019), to simple regularization terms in TD3+BC (Fujimoto & Gu, 2021), and more recently, advantage-weighted regression or diffusion-based modeling in IQL (Kostrikov et al., 2021) and DQL (Wang et al., 2022). In contrast, model-based methods leverage learned dynamics for data augmentation but must incorporate conservatism—often via uncertainty quantification or adversarial training—to prevent exploiting model errors in unreliable regions (Yu et al., 2020b; 2021; Rigter et al., 2022; Sun et al., 2023).

D Diffusion Background

Diffusion Model. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) generate samples by learning to invert a gradual noise-injection process. The framework is defined by a fixed forward process that progressively perturbs data $\boldsymbol{x_0}$ from the unknown data distribution $p_{\text{data}}(\boldsymbol{x_0})$ into Gaussian noise, governed by the transition:

$$\begin{aligned} q(\boldsymbol{x_{1:T}} \mid \boldsymbol{x_0}) &:= \prod_{t=1}^{T} q(\boldsymbol{x_t} \mid \boldsymbol{x_{t-1}}), \\ q(\boldsymbol{x_t} \mid \boldsymbol{x_{t-1}}) &:= \mathcal{N}(\boldsymbol{x_t}; \sqrt{1-\beta_t}\boldsymbol{x_{t-1}}, \beta_t \boldsymbol{I}), \end{aligned} \tag{19}$$

where T represents the total diffusion horizon and β_t regulates the noise variance schedule. The generative component corresponds to the reverse process $p_\theta(\boldsymbol{x_{0:T}}) := p(\boldsymbol{x_T}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}} \mid \boldsymbol{x_t})$, which is optimized by maximizing the Evidence Lower Bound (ELBO) (Blei et al., 2017): $\mathbb{E}_q \left[\log \frac{p_\theta(\boldsymbol{x_{0:T}})}{q(\boldsymbol{x_{1:T}} \mid \boldsymbol{x_0})} \right]$. Once trained, new data is synthesized by initializing $\boldsymbol{x_T} \sim p(\boldsymbol{x_T})$ and sequentially applying the learned denoising transitions to recover $\boldsymbol{x_0}$. This formulation can be naturally extended to conditional tasks by conditioning the reverse distribution on context c, denoted as $p_\theta(\boldsymbol{x_{t-1}} \mid \boldsymbol{x_t}, c)$.