EVALUATING OOD ACTIONS IN OFFLINE RL VIA QUANTILE ADVANTAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning must improve on recorded behavior while controlling errors in evaluating out-of-distribution (OOD) actions. We introduce Advantage-Based Diffusion Actor-Critic (ADAC), which compares candidate and behavior actions through their successor values. A learned dynamics model predicts these successors; a state-value function learned from offline transitions evaluates them. Subtracting a quantile of behavior-induced values from the candidate's successor value defines its quantile advantage. This signed signal corrects critic targets, guiding a diffusion actor alongside behavior cloning. The comparison covers in-distribution and OOD actions; its accuracy depends on the learned dynamics and values. We characterize reference-value updates and the augmented fixed-policy evaluation operator. ADAC achieves the highest reported mean on 15 of 20 D4RL tasks. Four-domain component ablations support the correction's contribution, and a 15-task quantile study examines its reference threshold. PointMaze illustrates shorter routes beyond the recorded patterns. Code: `https://anonymous.4open.science/r/adac-14D0`.

1 INTRODUCTION

Offline reinforcement learning (RL) learns from fixed data without further environment interaction (Lange et al., 2012; Levine et al., 2020; Prudencio et al., 2024). Improving on recorded behavior is difficult because bootstrapping can amplify overestimated values for out-of-distribution (OOD) actions (Fujimoto et al., 2019; Kumar et al., 2020). Existing safeguards include conservative value learning (Kumar et al., 2020; Kostrikov et al., 2021; Lyu et al., 2022), behavior constraints (Fujimoto & Gu, 2021; Fujimoto et al., 2019; Wang et al., 2022), and pessimistic model-based learning (Kidambi et al., 2020; Yu et al., 2020; Sun et al., 2023).

An unfamiliar action need not lead to an unfamiliar state. A state–action pair may be scarce in the dataset even when other recorded transitions reach its successor and reveal useful continuations from there. This distinction motivates evaluating OOD actions through their successors: the data may offer useful information about where an unfamiliar action leads.

Consider a navigation dataset of successful routes with long detours, as in our PointMaze study (Figure 4). An action that connects to a later part of a recorded route could avoid a detour while reusing a continuation already represented in the data. To assess such a candidate, we need to predict its successor and evaluate that continuation. When these estimates are reliable, comparing it with the successors of behavior actions provides a local reference for improvement.

This leads to our question: *does a candidate action lead to a more valuable continuation than the behavior actions available at the same state?* The same comparison applies to both in-distribution and OOD actions. Our approach complements ReFORM (Zhang et al., 2026), which optimizes the action distribution within the support of a learned behavior-cloning policy, by using predicted successor values to compare candidates with behavior.

We introduce **Advantage-Based Diffusion Actor-Critic (ADAC)** to make this comparison. A one-step dynamics model predicts each candidate's successor, and a state-value function learned from offline transitions evaluates the continuation. Applying the same models to behavior actions gives a reference distribution. Subtracting a quantile of this distribution from the candidate's successor value yields its *quantile advantage*. The quantile sets the threshold for positive correction.

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.

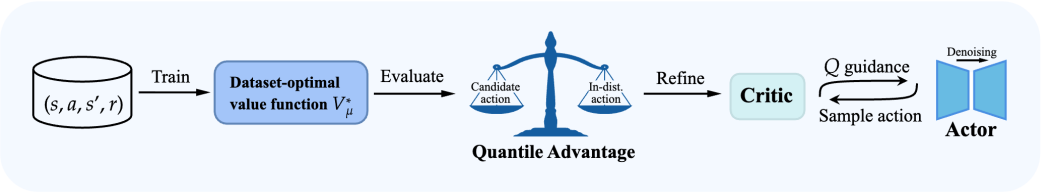


Figure 1: ADAC compares candidate and behavior-induced successor values, then uses quantile advantage to adjust critic targets and guide the diffusion actor.

Figure 1 shows how this comparison enters learning. Expectile regression learns the reference state value from offline transitions. Quantile advantage raises or lowers the critic's temporal-difference target according to the proposed next action's continuation, alongside the observed reward and estimated future return. A diffusion actor (Ho et al., 2020; Wang et al., 2022) learns through behavior cloning and guidance from this critic. ADAC retains DQL's actor objective (Wang et al., 2022) and adds a behavior-relative evaluation signal to critic learning. Behavior supplies both a cloning target and a reference for improvement.

Our central contribution is using behavior-induced successor values to guide critic learning. We characterize exact fitted expectile updates and establish contraction and perturbation bounds for the augmented fixed-policy evaluation operator. ADAC attains the highest reported mean on 15 of 20 D4RL tasks (Fu et al., 2020), and PointMaze shows shorter routes beyond recorded patterns. A four-domain component ablation and a 15-task quantile sweep separately examine the correction's benefit and its reference threshold.

2 Preliminaries

Offline Reinforcement Learning. RL problems are commonly formulated within the framework of a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \rho_0, P, \gamma)$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, and $r(\boldsymbol{s}, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$ is a bounded reward function with $R_{\max}$ being the maximum absolute value of the reward. $\rho_0(\boldsymbol{s})$ specifies the initial state distribution, $P(\boldsymbol{s'} \mid \boldsymbol{s}, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ defines the transition dynamics, and $\gamma \in (0, 1)$ is the discount factor (Sutton & Barto, 2018).

A policy $\pi(\cdot|\boldsymbol{s})$ maps a given state $\boldsymbol{s}$ to a probability distribution over the action space. The value function of a state $\boldsymbol{s}$ under a policy π is the expected cumulative return when starting in $\boldsymbol{s}$ and following π thereafter, i.e., $V^\pi(\boldsymbol{s}) = \mathbb{E}_{\boldsymbol{a_t} \sim \pi(\cdot|\boldsymbol{s_t})}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s_t}, \boldsymbol{a_t}) \,\middle|\, \boldsymbol{s_0} = \boldsymbol{s}\right]$, where the expectation takes over the randomness of the policy π and the transition dynamics P. The optimal state value function $V^*(\cdot)$ satisfies the following Bellman optimality equation:

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a} \in \mathcal{A}} \left[r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \mathbb{E}_{\boldsymbol{s'} \sim P(\cdot \mid \boldsymbol{s}, \boldsymbol{a})} V^*(\boldsymbol{s'}) \right]. \tag{1}$$

The action value is $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s_t}, \boldsymbol{a_t}) \mid \boldsymbol{s_0} = \boldsymbol{s}, \boldsymbol{a_0} = \boldsymbol{a}]$. The objective is

$$J(\pi) = \int_{\mathcal{S}} \rho_0(\boldsymbol{s}) V^\pi(\boldsymbol{s})\, ds = \mathbb{E}_{\boldsymbol{s_0} \sim \rho_0, \boldsymbol{a_t} \sim \pi, \boldsymbol{s_{t+1}} \sim P} \left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s_t}, \boldsymbol{a_t}) \right]. \tag{2}$$

Offline RL optimizes this objective using only a fixed dataset $\mathcal{D} = \{(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s'})\}$ collected by a behavior policy $\mu(\cdot|\boldsymbol{s})$, without additional environment interaction during training.

Expectile Regression. Expectile regression (Newey & Powell, 1987; Kostrikov et al., 2021) defines the τ-expectile, $\tau \in (0, 1)$, of a real-valued random variable x by asymmetric least squares:

$$\arg\min_{y \in \mathbb{R}} \mathbb{E}_x \left[L_2^\tau (x - y) \right], \tag{3}$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$. At $\tau = 0.5$ this gives the mean; $\tau > 0.5$ weights larger outcomes more heavily. As $\tau \to 1$, the expectile approaches the essential supremum of x.

3 EVALUATING ACTIONS THROUGH THEIR SUCCESSOR STATES

We connect successor-value comparison to critic and actor learning, then describe reference-value learning and characterize the corrected evaluation operator.

3.1 COMPARING CANDIDATE AND BEHAVIOR ACTIONS

Suppose we have a state-value function V that estimates the quality of continuations available in the offline data. At a state $\boldsymbol{s}$, we evaluate a candidate action $\boldsymbol{a}$ by the expected value of its successor. To put this value in context, we draw reference actions from the behavior policy $\mu(\cdot|\boldsymbol{s})$ and evaluate their successors in the same way. The candidate's *quantile advantage* is its successor value minus a quantile of these reference values:

$$A(\boldsymbol{a}|\boldsymbol{s}) := \mathbb{E}_{\boldsymbol{s'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a})} V(\boldsymbol{s'}) - \text{Quantile}_{\kappa}\left(\left\{\mathbb{E}_{\boldsymbol{s_i'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a_i})} V(\boldsymbol{s_i'})\right\}_{i=1}^{N}\right), \tag{4}$$

where $\{\boldsymbol{a_i}\}_{i=1}^{N}$ are independent samples from $\mu(\cdot|\boldsymbol{s})$. We use $N = 25$ reference actions. Here, the quantile summarizes variation across behavior actions at one state: each action contributes its expected successor value. The selected reference level sets the threshold for a positive candidate score.

A positive $A(\boldsymbol{a}|\boldsymbol{s})$ means that the candidate's expected successor value exceeds this reference level; a negative value means it falls below it. Increasing κ raises the threshold for a positive score, making the comparison more conservative. The score can be computed for both in-distribution and OOD candidates without first classifying their distributional status.

Here, "advantage" denotes relative successor value, distinct from $Q^{\pi}(s,a) - V^{\pi}(s)$. Actions with identical predicted successors receive the same score even if their immediate rewards differ; the observed reward remains in the critic target.

3.2 USING THE COMPARISON IN CRITIC UPDATES

We now use the relative score to adjust policy evaluation. At the next state $\boldsymbol{s'}$, a proposed action $\boldsymbol{a'}$ is compared with behavior actions available there. Its score is added to the next-action value in the Bellman target:

$$\mathcal{T}_A^{\pi_\theta} Q(\boldsymbol{s},\boldsymbol{a}) := r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\boldsymbol{s'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a}),\boldsymbol{a'}\sim\pi_\theta}\left[Q(\boldsymbol{s'},\boldsymbol{a'}) + \lambda A(\boldsymbol{a'}|\boldsymbol{s'})\right], \tag{5}$$

where $\lambda \geq 0$ controls the correction strength.

The correction raises the target for next actions above the reference threshold and lowers it for those below. The actor combines behavior cloning with guidance toward higher values under the corrected critic. Thus the local comparison affects the policy through critic learning. In practice we bound the correction with a sign-preserving soft-clipping function; Section 4 gives the network losses and updates.

For a shared reference set, the baseline preserves candidate ordering at a state while setting the boundary between positive and negative corrections. It cancels additive offsets shared by candidate and reference predictions, but not action-dependent errors. Section 3.4 explains how the resulting corrections accumulate along future continuations.

Figure 2 illustrates this motivation; Section 5.2 examines the resulting routes in PointMaze.

3.3 LEARNING THE REFERENCE STATE VALUE

The comparison requires a value function that reflects good continuations available in the data. Unlike Q, whose targets include learning-policy actions, V is fitted using observed rewards and successor states. We use expectile regression to approximate the following *dataset-optimal value function* (Lyu et al., 2022):

$$V_\mu^*(\boldsymbol{s}) := \max_{\boldsymbol{a}\sim\mu(\cdot|\boldsymbol{s})} \left[r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\boldsymbol{s'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a})}\left[V_\mu^*(\boldsymbol{s'})\right]\right]. \tag{6}$$

The maximization is restricted to behavior-supported actions. Throughout the analysis, maxima over random draws denote essential suprema under the indicated conditional distributions; for finite

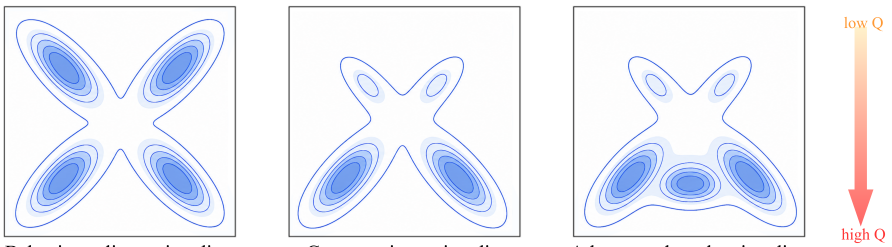


Figure 2: Conceptual illustration of advantage-based evaluation. Lower regions denote higher Q-values; the right panel illustrates the intended encouragement of valuable candidate actions beyond the behavior distribution.

supports they are ordinary maxima. We study population, statewise fitted regression and approximate it using offline samples. For a target function W held fixed within a regression step, the empirical objective is

$$\mathcal{L}(V; W) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s'})\sim\mathcal{D}}\big[L_2^\tau(r(\boldsymbol{s},\boldsymbol{a}) + \gamma W(\boldsymbol{s'}) - V(\boldsymbol{s}))\big]. \tag{7}$$

The following result concerns exact population updates over bounded measurable value functions, with conditional sampling law $\mu(\boldsymbol{a}|\boldsymbol{s})P(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})$.

Proposition 1. *For fixed W, the statewise regression solution is $\mathcal{T}_\tau W(\boldsymbol{s}) = \mathbb{E}^\tau_{\boldsymbol{a}\sim\mu,\boldsymbol{s'}\sim P}[r(\boldsymbol{s},\boldsymbol{a}) + \gamma W(\boldsymbol{s'})]$. The fitted iteration $V_{k+1} = \mathcal{T}_\tau V_k$ converges in the sup norm to the unique bounded fixed point*

$$V_\tau(\boldsymbol{s}) = \mathbb{E}^\tau_{\boldsymbol{a}\sim\mu(\cdot|\boldsymbol{s}),\ \boldsymbol{s'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a})}\big[r(\boldsymbol{s},\boldsymbol{a}) + \gamma V_\tau(\boldsymbol{s'})\big], \tag{8}$$

where $\mathbb{E}^\tau_x[x]$ denotes the τ-expectile of a real-valued random variable x.

Proposition 1 identifies the value learned by exact fitted updates. The next result explains its relation to the dataset-optimal value.

Proposition 2. *The fixed point V_τ in Proposition 1 satisfies $\|V_\tau\|_\infty \le R_{\max}/(1-\gamma)$ and is monotonically non-decreasing in τ. Furthermore, $V_\tau(\boldsymbol{s}) \to \bar{V}(\boldsymbol{s})$ pointwise as $\tau \to 1$, where $\bar{V}(\boldsymbol{s})$ is given by*

$$\bar{V}(\boldsymbol{s}) = \max_{a\sim\mu(\cdot|\boldsymbol{s})}\left[r(\boldsymbol{s},\boldsymbol{a}) + \gamma \max_{\boldsymbol{s'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a})} \bar{V}(\boldsymbol{s'})\right]. \tag{9}$$

In the case of deterministic transition dynamics, the limit coincides with the dataset-optimal value function Eq. (6), i.e., $\bar{V}(\boldsymbol{s}) = V^*_\mu(\boldsymbol{s})$.

Proposition 2 shows that for deterministic transition dynamics, $V_\tau(\boldsymbol{s})$ converges to $V^*_\mu(\boldsymbol{s})$ as $\tau \to 1$. For stochastic transitions, the limit maximizes over successor outcomes and can exceed V^*_μ; choosing $\tau < 1$ moderates this optimism but does not by itself guarantee an accurate approximation. These results characterize exact fitted updates, rather than convergence of finite-sample neural-network optimization. The two parameters play distinct roles: τ controls value fitting, whereas κ in Eq. (4) sets the behavior reference for action comparison.

3.4 Properties of the Corrected Evaluation Operator

We now characterize the operator in Eq. (5) with the policy and comparison held fixed.

Proposition 3. *For a fixed policy π_θ and a fixed bounded advantage function A, $\mathcal{T}_A^{\pi_\theta}$ is a γ-contraction on bounded Q-functions in the sup norm and has a unique fixed point.*

Proposition 3 ensures a unique evaluation target for fixed π_θ and A. Writing this target as $Q^A_{\pi_\theta}$ and the ordinary action value as Q_{π_θ}, we next bound the change caused by the correction.

Proposition 4. *Under Proposition 3, if $\lambda \ge 0$ and the value function defining A satisfies $\|V\|_\infty \le R_{\max}/(1-\gamma)$ (as does V_τ), then*

$$\|Q^A_{\pi_\theta} - Q_{\pi_\theta}\|_\infty \le 2\lambda R_{\max}(1-\gamma)^{-2}.$$

Algorithm 1 Advantage-Based Diffusion Actor-Critic

1: **Input** Dataset $\mathcal{D}$, policy network π_θ, critic networks Q_ϕ
2: Train a behavior policy μ by minimizing Eq. (13)
3: Train a value function V by minimizing Eq. (11)
4: Train a transition model P by minimizing Eq. (12)
5: **for** each iteration **do**
6: Sample a batch $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s'})$ from $\mathcal{D}$ and next actions $\boldsymbol{a'} \sim \pi_\theta(\cdot|\boldsymbol{s'})$
7: Obtain $A(\boldsymbol{a'}|\boldsymbol{s'})$ according to Eq. (4)
8: Update $\mathcal{L}_{\mathrm{CRITIC}}(\phi)$ according to Eq. (14)
9: Update $\mathcal{L}_{\mathrm{ACTOR}}(\theta)$ according to Eq. (15)
10: Soft update target-network parameters
11: **end for**

To see what this fixed point represents, let $B = \lambda A$, or $B = \mathrm{softclip}(A)$ for the bounded implementation. For fixed π and B, unrolling the evaluation equation gives

$$Q_\pi^B(s,a) = Q_\pi(s,a) + \mathbb{E}_\pi\left[\sum_{t=1}^{\infty} \gamma^t B(s_t,a_t) \,\middle|\, s_0 = s, a_0 = a\right]. \tag{10}$$

The critic therefore accumulates the comparison along future continuations. A state-dependent reference can change these accumulated corrections even though its subtraction preserves the local ordering of candidate scores. Appendix B derives this identity; it characterizes the modified evaluation objective, not a guarantee of improved environment return.

4 Advantage-Based Diffusion Actor-Critic Algorithm

ADAC implements the comparison and critic correction from Section 3 with learned networks. We describe the models that compute the score, the diffusion policy, and the actor–critic updates in that order.

Computing the Quantile Advantage. In our practical implementation, the value function is parameterized by a neural network with parameters φ and trained by minimizing the following expectile regression loss:

$$\mathcal{L}_{\mathrm{VALUE}}(\varphi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s'})\sim\mathcal{D}}\big[L_2^\tau(r + \gamma V_\varphi(\boldsymbol{s'}) - V_\varphi(\boldsymbol{s}))\big]. \tag{11}$$

We use a deterministic neural-network transition model to predict the successor state for this comparison. The model is trained by minimizing the following mean squared error (MSE) loss:

$$\mathcal{L}_{\mathrm{MODEL}}(\psi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})\sim\mathcal{D}}\big[\|P_\psi(\boldsymbol{s},\boldsymbol{a}) - \boldsymbol{s'}\|^2\big]. \tag{12}$$

The behavior policy μ, reference value V, and transition model P are learned from offline data using Eq. (13), Eq. (11), and Eq. (12), respectively, then kept fixed during actor–critic updates. To compute Eq. (4), we evaluate $V_\varphi(P_\psi(\boldsymbol{s},\boldsymbol{a}))$ for the candidate and each reference action. For stochastic dynamics, this uses a point prediction to approximate the expected successor value.

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

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

where $\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s})$ is the marginal of the terminal sample $\boldsymbol{a^0}$. With $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{j=1}^{i} \alpha_j$, the DDPM noise-prediction objective on dataset pairs $(\boldsymbol{s},\boldsymbol{a})$ is

$$\mathcal{L}_{\mathrm{BC}}(\theta) = \mathbb{E}_{i\sim\mathcal{U},\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}),(\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_i}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}, \boldsymbol{s}, i)\big\|^2\Big], \tag{13}$$

where $\mathcal{U}$ is uniform over $\{1,\dots,T\}$ and $\boldsymbol{\epsilon}_\theta$ predicts the added noise. This behavior-cloning (BC) loss fits the action distribution in the dataset. Actions are generated by reversing the diffusion from Gaussian noise; Appendix D gives the forward process.

Actor-Critic. The critic fits the corrected Bellman target from Eq. (5) using dataset transitions:

$$\begin{aligned}\mathcal{L}_{\mathrm{CRITIC}}(\phi) =& \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})\sim\mathcal{D},\boldsymbol{a'}\sim\pi_\theta(\cdot|\boldsymbol{s'})}[(r(\boldsymbol{s},\boldsymbol{a}) \\ &+ \gamma\big(Q_\phi(\boldsymbol{s'},\boldsymbol{a'}) + \lambda A(\boldsymbol{a'} \mid \boldsymbol{s'})\big) - Q_\phi(\boldsymbol{s},\boldsymbol{a}))^2].\end{aligned} \tag{14}$$

The implementation replaces λA with the bounded soft-clipping transformation in Appendix C.1. It preserves the correction's sign and compresses large magnitudes, so the behavior reference still separates positive from negative corrections. The models defining A remain fixed. Predicted successors are used to compute this comparison; the critic is fitted on observed dataset transitions.

The actor combines behavior cloning with guidance from the corrected critic, encouraging actions with higher estimated values:

$$\mathcal{L}_{\mathrm{ACTOR}}(\theta) = \mathcal{L}_{\mathrm{BC}}(\theta) - \alpha\mathbb{E}_{\boldsymbol{s}\sim\mathcal{D},\boldsymbol{a}\sim\pi_\theta}\left[Q_\phi(\boldsymbol{s},\boldsymbol{a})\right]. \tag{15}$$

Algorithm 1 summarizes how the auxiliary comparison enters an otherwise diffusion-based actor–critic procedure. The correction is applied to the proposed next action in the critic target; it does not serve as a regression weight in the actor's cloning loss.

5 Experiments

We evaluate ADAC from three perspectives: policy returns on D4RL (Fu et al., 2020), route generalization in PointMaze, and the contribution of quantile advantage. The component ablation tests the correction's effect on performance, while the κ study examines how its reference threshold affects returns. We then report computational cost.

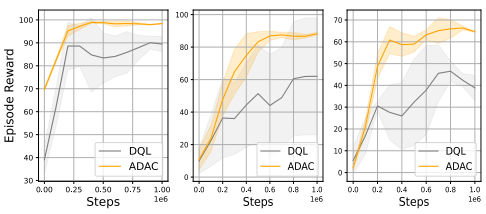
Datasets. We evaluate 20 D4RL tasks across four domains. Gym-MuJoCo tests locomotion with dense rewards, and AntMaze tests navigation with sparse rewards and suboptimal trajectories. Adroit includes human demonstrations and cloned-policy data for dexterous manipulation. Kitchen tests completion of multiple manipulation subtasks from demonstrations. Together, these domains cover different reward structures and sources of offline behavior.

Baselines. We compare against policy regularization, value-learning, and generative approaches. Policy regularization baselines include BC, TD3+BC (Fujimoto & Gu, 2021), BEAR (Kumar et al., 2019), BRAC (Wu et al., 2019), BCQ (Fujimoto et al., 2019), and AWR (Peng et al., 2019). For value-learning approaches, we include CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021), REM (Agarwal et al., 2020), AlignIQL (He et al., 2024), and IDQL-A (Hansen-Estruch et al., 2023). For generative-policy and trajectory-modeling approaches, we include Diffuser (Janner et al., 2022), DQL (Wang et al., 2022), DTQL (Chen et al., 2024), SfBC (Chen et al., 2022), and DAC (Fang et al., 2024). We report the performance of baseline methods either from the best results published in their respective papers or reported by Wang et al. (2022).

5.1 Policy Performance on D4RL

Table 1 reports policy returns across the four domains. These results evaluate the complete algorithm; Section 5.3 examines the contribution of the correction itself.

Results for AntMaze Tasks. We follow the D4RL evaluation protocol with *normalized* scores, where 100 corresponds to an expert policy (Fu et al., 2020). AntMaze tests learning from sparse rewards and suboptimal trajectories, a setting in which comparing the outcomes of candidate actions is particularly relevant to our motivation. ADAC obtains a domain-average score of 81.7, compared with 69.6 for DQL and 79.1 for IDQL-A, the strongest listed baseline by domain average. On Umaze, Medium Diverse, and Large Play (Figure 3), ADAC also has higher mean returns for most of training and generally narrower standard-deviation bands than DQL. These curves complement the final-score comparison in this sparse-reward setting.



(a) Umaze (b) Medium-Diverse (c) Large-Play

Figure 3: AntMaze returns over 4 seeds (mean ± std.; 10-point smoothing).

Table 1: Normalized returns on D4RL tasks. ADAC averages the final 10 evaluations across 4 seeds; baseline means come from the cited sources. Dark and light shading mark the highest and second-highest reported means, respectively.

Gym Tasks	**BC**	**CQL**	**IQL**	**Diffuser**	**DTQL**	**AlignIQL**	**SfBC**	**DQL**	**IDQL-A**	**ADAC (Ours)**
halfcheetah-medium	42.6	44.0	47.4	44.2	57.9	46.0	45.9	51.1	51.0	58.0±0.3
hopper-medium	52.9	58.5	66.3	58.5	99.6	56.1	57.1	90.5	65.4	93.5±4.2
walker2d-medium	75.3	72.5	78.3	79.7	89.4	78.5	77.9	87.0	82.5	87.6±2.0
halfcheetah-medium-replay	36.6	45.5	44.2	42.2	50.9	41.1	37.1	47.8	45.9	52.5±0.8
hopper-medium-replay	18.1	95.0	94.7	96.8	100.0	74.8	86.2	101.3	92.1	102.1±1.1
walker2d-medium-replay	26.0	77.2	73.9	61.2	88.5	76.5	65.1	95.5	85.1	96.0±1.6
halfcheetah-medium-expert	55.2	91.6	86.7	79.8	92.7	89.1	92.6	96.8	95.9	106.1±1.0
hopper-medium-expert	52.5	105.4	91.5	107.2	109.3	107.1	108.6	111.1	108.6	112.5±1.0
walker2d-medium-expert	107.5	108.8	109.6	108.4	110.0	111.9	109.8	110.1	112.7	112.9±0.7
Average	51.9	77.6	77.0	75.3	88.7	75.7	75.6	87.9	82.1	91.2
AntMaze Tasks	**BC**	**TD3+BC**	**CQL**	**IQL**	**DTQL**	**AlignIQL**	**SfBC**	**DQL**	**IDQL-A**	**ADAC (Ours)**
antmaze-umaze	54.6	78.6	74.0	87.5	94.8	94.8	92.0	93.4	94.0	98.2±4.5
antmaze-umaze-diverse	45.6	71.4	84.0	62.2	78.8	82.4	85.3	66.2	80.2	82.5±4.9
antmaze-medium-play	0.0	10.6	61.2	71.2	79.6	80.5	81.3	76.6	84.2	86.5±9.8
antmaze-medium-diverse	0.0	3.0	53.7	70.0	82.2	85.5	82.0	78.6	84.8	88.7±10.2
antmaze-large-play	0.0	0.2	15.8	39.6	52.0	65.2	59.3	46.4	63.5	69.8±12.4
antmaze-large-diverse	0.0	0.0	14.9	47.5	66.4	54.0	45.5	56.6	67.9	64.6±12.7
Average	16.7	27.3	50.6	63.0	75.6	77.1	74.2	69.6	79.1	81.7
Adroit Tasks	**BC**	**BRAC-v**	**CQL**	**IQL**	**BEAR**	**REM**	**BCQ**	**DAC**	**DQL**	**ADAC (Ours)**
pen-human	25.8	0.6	35.2	71.5	-1.0	5.4	68.9	81.3	72.8	74.4±18.6
pen-cloned	38.3	-2.5	27.2	37.3	26.5	-1.0	44.0	63.9	57.3	80.5±14.3
Average	32.1	-1.0	31.2	54.4	12.8	2.2	56.5	72.6	65.1	77.5
Kitchen Tasks	**BC**	**BRAC-v**	**CQL**	**IQL**	**BEAR**	**AWR**	**BCQ**	**DAC**	**DQL**	**ADAC (Ours)**
kitchen-complete	33.8	0.0	43.8	62.5	0.0	0.0	8.1	77.4	84.0	87.9±6.7
kitchen-partial	33.8	0.0	49.8	46.3	13.1	15.4	18.9	50.0	60.5	65.2±7.0
kitchen-mixed	47.5	0.0	51.0	51.0	47.2	10.6	8.1	60.2	62.6	68.3±5.8
Average	38.4	0.0	48.2	53.3	20.1	8.7	11.7	62.6	69.0	73.8

Results for Gym Tasks. With dense rewards, ADAC obtains a domain-average score of 91.2, compared with 87.9 for DQL and 88.7 for DTQL, the strongest listed baseline by domain average. ADAC has the highest reported mean on seven of nine tasks; DTQL leads on Hopper-medium and Walker2d-medium.

Results for Adroit and Kitchen Tasks. The gains extend to manipulation: ADAC averages 77.5 on Adroit, compared with 65.1 for DQL and 72.6 for DAC. On Kitchen, it has the highest reported mean on all three tasks. The following studies examine the learned routes and the contribution of the correction.

5.2 Route Generalization in PointMaze

PointMaze makes the introduction's navigation example concrete: can ADAC learn shorter routes from successful but circuitous trajectories? We use a sparse-reward environment built with `gymnasium-robotics` (de Lazcano et al., 2023), based on D4RL Maze2D. The dataset contains 853 trajectories and 391,391 transitions following three route patterns (Figure 4). Both ADAC and DQL train for 50 000 steps and are evaluated using 300 trajectories each. Appendix C.6 gives the environment and data-collection details.

The offline data contain no trajectories shorter than 175 steps. ADAC nevertheless generates short, direct routes absent from the dataset, whereas DQL more closely follows its left, middle, and right route patterns (Figure 4). The reported length distributions place 37.5% of ADAC trajectories below

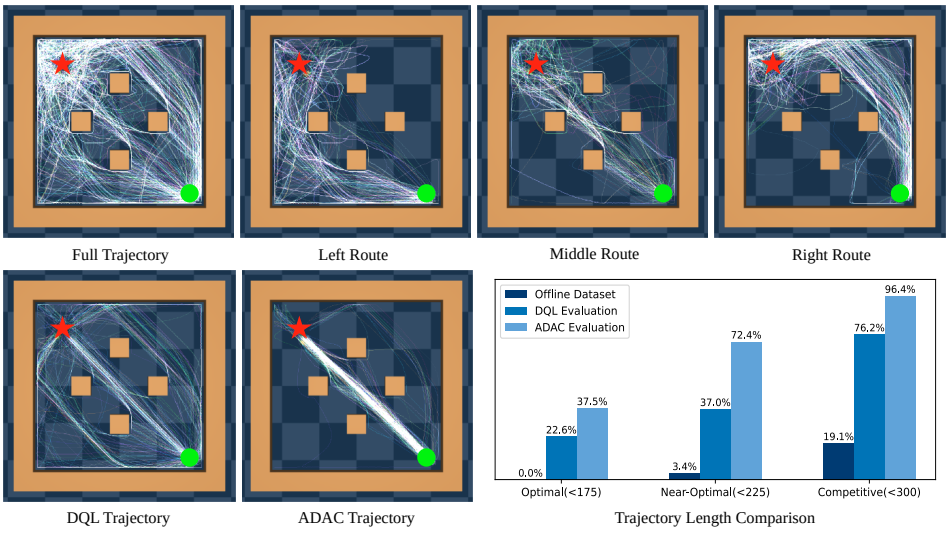


Figure 4: PointMaze routes and trajectory lengths. Top: the offline dataset and its three route patterns. Bottom: DQL and ADAC routes, with cumulative trajectory-length proportions.

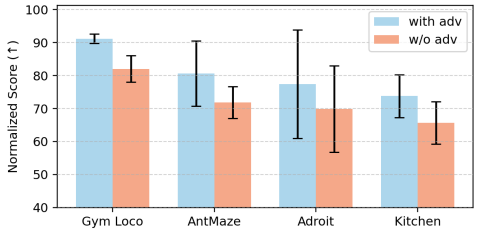


Figure 5: Quantile-advantage ablation: mean normalized scores across four domains.

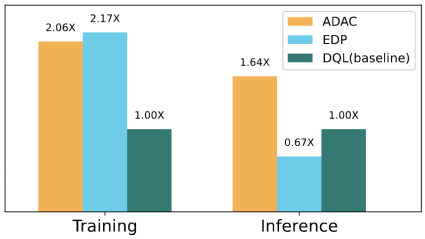


Figure 6: Training and inference speedups relative to DQL, with EDP for comparison.

175 steps, compared with 22.6% for DQL; below 300 steps, the proportions are 96.4% and 76.2%, respectively. Together with the route plots, these results show generalization beyond the recorded route patterns. They do not establish action-level OOD status: behavior-supported actions can also compose a new route.

5.3 Ablation Study

Quantile advantage. Figure 5 compares ADAC with and without the correction. Adding quantile advantage yields relative improvements of **11.1%** on Gym Locomotion, **12.2%** on AntMaze, **10.9%** on Adroit, and **12.4%** on Kitchen. The gains across all four domains support the correction's contribution beyond the sparse-reward navigation setting that motivates our method.

For fixed candidate and reference actions, lowering κ makes positive corrections easier to obtain without changing action ordering. These corrections accumulate through the critic (Eq. (10)), so we next examine how the reference threshold affects policy returns.

Reference quantile. Figure 7 varies $\kappa \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$ on 15 tasks, with four seeds per setting and $N = 25$ reference actions. In AntMaze, $\kappa = 0.95$ sharply reduces returns on Medium Play and both Large tasks; lower or intermediate thresholds perform better there. Gym Locomotion is more task-dependent: Hopper medium-replay and medium-expert benefit from higher thresholds, while most HalfCheetah and Walker2d settings vary less. The results support task-specific reference levels rather than a universally best κ.

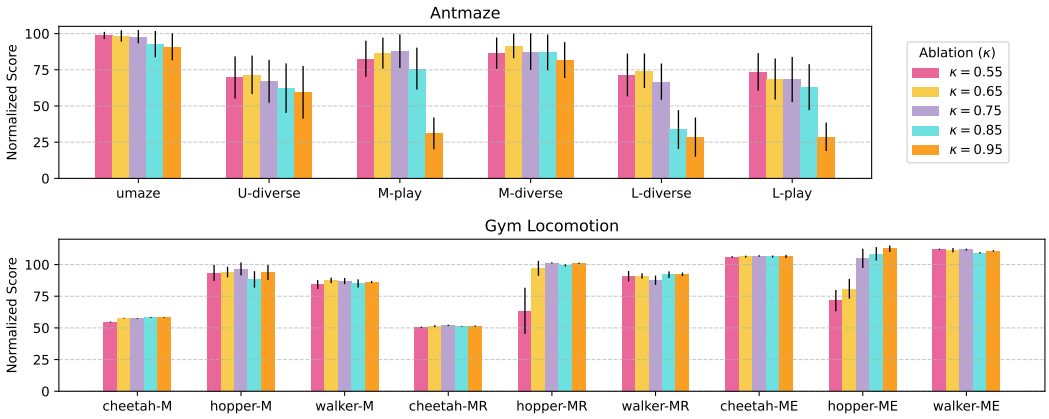


Figure 7: Sensitivity to κ on six AntMaze and nine Gym tasks (4 seeds; $N = 25$ reference actions). Larger κ raises the threshold for positive correction.

Taken together, the component ablation and the quantile sweep separate the benefit of the correction from the choice of its reference level. The former tests whether adding the signal improves policy returns; the latter shows how those returns depend on the threshold. The reported gains therefore correspond to the task-specific settings in Appendix C.4, rather than a single quantile shared by all tasks.

Correction statistics. Appendix C.1 summarizes the quantile advantage over four runs on each of the 20 D4RL tasks under its best-performing hyperparameter setting. Both positive and negative values occur in every task. The positive fraction ranges from 10.1% on `hopper-medium-expert` to 62.3% on `antmaze-large-diverse`. These fractions count candidates whose predicted successor values exceed the behavior reference; they do not measure the fraction of truly improving or OOD actions. The statistics describe how the signal is distributed, complementing the return-based ablation that tests whether using it benefits the learned policy.

5.4 Computational Efficiency

On one NVIDIA RTX 4090, our `jaxrl_m`-based implementation achieves training and inference speedups of $2\times$ and $1.64\times$, respectively, over original DQL (Figure 6). Training throughput is comparable to EDP (Kang et al., 2023), with faster inference. These end-to-end comparisons include the JAX reimplementation and do not isolate the correction's cost.

Auxiliary pretraining is reported separately. The transition and behavior models each train for 300 000 gradient steps and each take under 10 minutes on the same GPU (Appendix C.5). These two models, together with the learned reference value, remain fixed during actor–critic training. The pretrained transition and behavior models can be reused across runs when their dataset and training configuration are unchanged. This separates the initial model-fitting cost from the subsequent policy-learning stage.

6 Conclusion

ADAC evaluates candidate actions, including OOD actions, through their successor values relative to a behavior-induced reference. Quantile advantage turns this comparison into a signed critic-target correction, connecting action evaluation to policy learning. The comparison relies on learned dynamics and values; our theory characterizes reference fitting and fixed-policy evaluation. Empirically, ADAC attains the highest reported mean on 15 of 20 D4RL tasks and learns shorter routes in PointMaze. Component ablations support the correction's contribution, while the quantile study shows how the reference threshold affects returns. These results illustrate a useful role for offline data: behavior provides both actions to imitate and continuations against which to evaluate new candidates.

Ethics Statement

Offline reinforcement learning has potential applications in domains such as robotics, autonomous systems, and industrial control, where online interaction may be costly or unsafe. By reducing the need for real-world exploration, offline RL may help improve data efficiency and deployment safety. However, methods that generalize beyond observed data may still produce unreliable behaviors under distributional shift or insufficient data coverage. Therefore, careful evaluation and appropriate safety constraints remain important before deploying such systems in real-world decision-making scenarios.

AI Use Statement

An AI assistant helped migrate the manuscript to the ICLR template, edit the language and structure, check compilation and layout, revise mathematical statements and proofs, and design supplementary experimental protocols. The authors are responsible for verifying the experimental results and all AI-assisted content.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pp. 104–114. PMLR, 2020.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. In *Advances in Neural Information Processing Systems*, volume 34, pp. 7436–7447, 2021.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

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.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097, 2021.

Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. Diffusion policies creating a trust region for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50098–50125, 2024.

Xuyang Chen, Guojian Wang, Keyu Yan, and Lin Zhao. VIPO: Value function inconsistency penalized offline reinforcement learning. *arXiv preprint arXiv:2504.11944*, 2025. URL https://arxiv.org/abs/2504.11944v1.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium Robotics. Software, 2023. URL https://github.com/Farama-Foundation/Gymnasium-Robotics.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bing-Yi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. *arXiv preprint arXiv:2405.20555*, 2024.

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. In *Advances in Neural Information Processing Systems*, volume 34, pp. 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.

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

Longxiang He, Li Shen, and Xueqian Wang. AlignIQL: Policy alignment in implicit Q-learning through constrained optimization. *arXiv preprint arXiv:2405.18187*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 67195–67212, 2023.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823, 2020.

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. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 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.

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.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1711–1724, 2022.

Whitney K Newey and James L Powell. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, pp. 819–847, 1987.

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

Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10237–10257, 2024. doi: 10.1109/TNNLS.2023.3250269.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for dynamics and control*, pp. 1154–1168. PMLR, 2021.

Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: Robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16082–16097, 2022.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. COG: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

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.

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.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. In *Advances in Neural Information Processing Systems*, volume 36, pp. 18532–18550, 2023.

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

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 28954–28967, 2021.

Songyuan Zhang, Oswin So, H. M. Sabbir Ahmad, Eric Yang Yu, Matthew Cleaveland, Mitchell Black, and Chuchu Fan. ReFORM: Reflected flows for on-support offline RL via noise manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YvFsyRReeN.

A Related Work

Offline RL learns policies from previously collected data, with applications explored in prior work (Rafailov et al., 2021; Singh et al., 2020; Li et al., 2010). The following groups describe common approaches; they are not mutually exclusive:

Pessimistic value-based methods achieve conservatism by incorporating penalty terms into the value optimization objective, discouraging the value function from being overly optimistic on out-of-distribution (OOD) actions. Specifically, CQL (Kumar et al., 2020) augments Bellman-error minimization with a conservative Q-value regularizer, whereas EDAC (An et al., 2021) and PBRL (Bai et al., 2022) adjust the penalization based on the uncertainty level of the Q-value, measured using a neural network ensemble.

Regularized policy-based methods constrain or regularize the learned policy toward the behavior data. BEAR (Kumar et al., 2019) uses an MMD constraint, BCQ (Fujimoto et al., 2019) generates candidates with a CVAE behavior model, and TD3+BC (Fujimoto & Gu, 2021) adds a cloning term to policy optimization. IQL (Kostrikov et al., 2021) fits values using dataset actions and extracts a policy through advantage-weighted behavior cloning. ADAC uses a different comparison and insertion point: predicted successor values relative to behavior-induced values supply a signed critic-target correction, while the actor retains its cloning objective.

Diffusion and flow policies provide expressive action distributions. DQL (Wang et al., 2022) combines diffusion behavior cloning with direct Q-function guidance. SfBC (Chen et al., 2022) separates behavior generation from action evaluation and selects among behavior-model candidates using value-based weights. ReFORM (Zhang et al., 2026) optimizes a latent distribution within the support of a learned behavior-cloning flow policy. ADAC builds on diffusion actor–critic learning and changes its critic target using a separate successor-value comparison. The behavior samples define the comparison reference, rather than an explicit constraint on the actor's support.

Conditional sequence modeling methods learn from offline trajectories through conditional prediction objectives (Chen et al., 2021; Wu et al., 2023). This leads to a supervised learning paradigm. Additionally, trajectories can be formulated as conditioned generative models and generated by diffusion models that satisfy conditioned constraints (Janner et al., 2022; Ajay et al., 2022).

Model-based methods incorporate conservatism to prevent the policy from overgeneralizing to regions where the dynamics model predictions are unreliable. For example, COMBO (Yu et al., 2021) extends CQL to a model-based setting by enforcing small Q-values for OOD samples generated by the dynamics model. RAMBO (Rigter et al., 2022) incorporates conservatism by adversarially training the dynamics model to minimize the value function while maintaining accurate transition predictions. Several model-based methods use uncertainty penalties to discourage exploiting unreliable model predictions. Specifically, MOPO (Yu et al., 2020) uses the max-aleatoric uncertainty quantifier, MOReL (Kidambi et al., 2020) employs the max-pairwise-diff uncertainty quantifier, and MOBILE (Sun et al., 2023) leverages the Model-Bellman inconsistency uncertainty quantifier. VIPO (Chen et al., 2025) uses inconsistency between data-derived and model-derived values to train the dynamics model. ADAC uses its dynamics model for a one-step action comparison and applies the resulting signal to critic learning; the model is fitted by transition regression.

B Proof of Propositions

We analyze exact population fitted updates on bounded measurable functions. Conditional distributions are those of the fixed behavior policy and the MDP. All sup norms are over the full domain under analysis; essential suprema of random targets are with respect to the stated conditional laws.

Proof of Proposition 1.

Proof. Fix a state $\boldsymbol{s}$ and a bounded target function W. The conditional target $X_W = r(\boldsymbol{s}, \boldsymbol{A}) + \gamma W(\boldsymbol{S'})$ is fixed with respect to the scalar prediction v. By the definition of expectile regression,

$$\arg\min_{v\in\mathbb{R}} \mathbb{E}\big[L_2^\tau(X_W - v)\big] = \mathbb{E}^\tau[X_W] =: \mathcal{T}_\tau W(\boldsymbol{s}).$$

Equivalently, its first-order condition is $\mathbb{E}[|\tau - \mathbb{1}(X_W - v < 0)|(X_W - v)] = 0$. The target W is held fixed in this minimization. The statement is therefore about fitted regression, not joint minimization through both occurrences of V in a Bellman residual.

For completeness, expectiles are monotone and translation-equivariant. Indeed, the function $h_X(v) = \tau\mathbb{E}(X - v)_+ - (1 - \tau)\mathbb{E}(v - X)_+$ is continuous and strictly decreasing in v, with its unique zero at $\mathbb{E}^\tau[X]$. If $X \le Y$ almost surely, then $h_X(v) \le h_Y(v)$ and hence $\mathbb{E}^\tau[X] \le \mathbb{E}^\tau[Y]$. Replacing X by $X + c$ shifts the zero by c. Consequently, if $|X - Y| \le c$ almost surely, then

$$\left|\mathbb{E}^\tau[X] - \mathbb{E}^\tau[Y]\right| \le c.$$

Coupling the targets using the same $(\boldsymbol{A}, \boldsymbol{S'})$ gives

$$\|\mathcal{T}_\tau V - \mathcal{T}_\tau W\|_\infty \le \gamma\|V - W\|_\infty.$$

The operator maps bounded measurable functions to bounded measurable functions. This space is complete in the sup norm, so the contraction mapping theorem gives a unique bounded fixed point V_τ and sup-norm convergence of $V_{k+1} = \mathcal{T}_\tau V_k$ from every bounded initial value. Its fixed-point equation is Eq. (8). □

Proof of Proposition 2.

Proof. Since an expectile lies between the essential infimum and supremum of its target,

$$\|\mathcal{T}_\tau V\|_\infty \le R_{\max} + \gamma\|V\|_\infty.$$

Applying this inequality at the fixed point yields $\|V_\tau\|_\infty \le R_{\max}/(1 - \gamma)$, uniformly in τ. For fixed X and v, the function $h_X(v)$ used above is nondecreasing in τ, since its derivative with respect to τ is $\mathbb{E}|X - v| \ge 0$. Its zero, the expectile, is therefore nondecreasing in τ. Together with monotonicity in the target, this implies that, for $\tau_1 \le \tau_2$, fitted iterations from a common bounded initial value remain ordered. Taking their limits gives $V_{\tau_1} \le V_{\tau_2}$. Thus there exists a bounded pointwise limit $\bar{V} = \lim_{\tau\uparrow 1} V_\tau$.

We now identify this limit without assuming uniform convergence in the state variable. Fix $\boldsymbol{s}$ and define

$$X_\tau = r(\boldsymbol{s}, \boldsymbol{A}) + \gamma V_\tau(\boldsymbol{S'}), \qquad \bar{X} = r(\boldsymbol{s}, \boldsymbol{A}) + \gamma\bar{V}(\boldsymbol{S'}),$$

under $(\boldsymbol{A}, \boldsymbol{S'}) \sim \mu(\cdot|\boldsymbol{s})P(\cdot|\boldsymbol{s}, \boldsymbol{A})$. These bounded targets satisfy $X_\tau \uparrow \bar{X}$. For any fixed $\eta \in (0, 1)$ and every $\tau \ge \eta$, monotonicity gives

$$\mathbb{E}^\tau[X_\eta] \le \mathbb{E}^\tau[X_\tau] = V_\tau(\boldsymbol{s}) \le \operatorname{ess\,sup} \bar{X}.$$

For a fixed bounded random variable X, its expectile tends to $\operatorname{ess\,sup} X$ as $\tau \uparrow 1$ (Kostrikov et al., 2021, Lemma 1). One can also see this directly from $\tau\mathbb{E}(X - m_\tau)_+ = (1 - \tau)\mathbb{E}(m_\tau - X)_+$: if the increasing limit of m_τ were below $\operatorname{ess\,sup} X$, the left-hand side would remain positive while the right-hand side tends to zero. Applying this fact to the *fixed* target X_η gives

$$\operatorname{ess\,sup} X_\eta \le \bar{V}(\boldsymbol{s}) \le \operatorname{ess\,sup} \bar{X}.$$

Take a sequence $\eta_n \uparrow 1$. For increasing random variables $X_{\eta_n} \uparrow \bar{X}$, their essential suprema increase to $\operatorname{ess\,sup} \bar{X}$: any common almost-sure upper bound for the countable sequence also bounds its pointwise limit almost surely. Hence

$$\begin{aligned}\bar{V}(\boldsymbol{s}) &= \operatorname*{ess\,sup}_{\boldsymbol{a}\sim\mu,\, \boldsymbol{s'}\sim P} \left[r(\boldsymbol{s}, \boldsymbol{a}) + \gamma\bar{V}(\boldsymbol{s'})\right] \\ &= \operatorname*{ess\,sup}_{\boldsymbol{a}\sim\mu(\cdot|\boldsymbol{s})} \left[r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \operatorname*{ess\,sup}_{\boldsymbol{s'}\sim P(\cdot|\boldsymbol{s},\boldsymbol{a})} \bar{V}(\boldsymbol{s'})\right],\end{aligned}$$

which is Eq. (9) under the stated maximum convention. For deterministic transitions the inner essential supremum equals the conditional expectation. Thus $\bar{V}$ is a fixed point of the behavior-restricted optimal Bellman operator in Eq. (6). That operator is a γ-contraction on bounded functions, so its fixed point is unique and $\bar{V} = V^*_\mu$. □

Proof of Proposition 3.

Proof. Fix π_θ and bounded A, and take two bounded functions Q_1, Q_2. The common advantage term cancels, giving

$$\begin{aligned}\|\mathcal{T}_A^{\pi_\theta} Q_1 - \mathcal{T}_A^{\pi_\theta} Q_2\|_\infty &= \gamma \sup_{\boldsymbol{s},\boldsymbol{a}} |\mathbb{E}_{\boldsymbol{s'}\sim P,\boldsymbol{a'}\sim\pi_\theta}[Q_1(\boldsymbol{s'},\boldsymbol{a'}) - Q_2(\boldsymbol{s'},\boldsymbol{a'})]| \\ &\leq \gamma \|Q_1 - Q_2\|_\infty.\end{aligned}$$

Bounded rewards and advantage ensure that the operator maps bounded functions to bounded functions. The contraction mapping theorem therefore gives its unique bounded fixed point. This is a fixed-policy, fixed-advantage statement. If the reference actions used for the empirical quantile are resampled, the corresponding expected operator uses the expected correction; the same argument applies when that correction is bounded and fixed in distribution. □

Proof of Proposition 4.

Proof. Let $M = R_{\max}/(1-\gamma)$ and assume $\|V\|_\infty \leq M$. Each conditional successor value and its empirical quantile lie in $[-M, M]$, so $\|A\|_\infty \leq 2M$. Subtracting the fixed-point equations for $Q^A_{\pi_\theta}$ and Q_{π_θ} gives

$$\|Q^A_{\pi_\theta} - Q_{\pi_\theta}\|_\infty \leq \gamma \|Q^A_{\pi_\theta} - Q_{\pi_\theta}\|_\infty + \gamma\lambda\|A\|_\infty.$$

Rearranging yields the slightly stronger bound

$$\|Q^A_{\pi_\theta} - Q_{\pi_\theta}\|_\infty \leq \frac{\gamma\lambda\|A\|_\infty}{1-\gamma} \leq \frac{2\gamma\lambda R_{\max}}{(1-\gamma)^2} \leq \frac{2\lambda R_{\max}}{(1-\gamma)^2}.$$

The assumed value bound holds for the exact V_τ by Proposition 2; it is not an automatic bound on an arbitrary learned neural value function. More generally, if a fixed bounded correction B replaces λA in the operator, the same argument gives $\|Q^B_{\pi_\theta} - Q_{\pi_\theta}\|_\infty \leq \gamma\|B\|_\infty/(1-\gamma)$. For the soft-clipping rule in Appendix C.1, $\|B\|_\infty \leq \max(\lambda_p, \lambda_n)$, providing a bound directly in terms of its saturation levels. □

Interpretation of the accumulated correction. Let B be a fixed bounded correction, and let Q^B_π denote the fixed point of the operator obtained by replacing λA with B in Eq. (5). Subtracting the ordinary policy-evaluation equation gives, with $\Delta = Q^B_\pi - Q_\pi$,

$$\begin{aligned}\Delta(s_0, a_0) &= \gamma\mathbb{E}_\pi[B(s_1, a_1) + \Delta(s_1, a_1) \mid s_0, a_0] \\ &= \mathbb{E}_\pi\left[\sum_{t=1}^{H} \gamma^t B(s_t, a_t) + \gamma^H \Delta(s_H, a_H) \,\middle|\, s_0, a_0\right].\end{aligned}$$

Both fixed points are bounded, so the remainder is bounded in absolute value by $\gamma^H\|\Delta\|_\infty$ and vanishes as $H \to \infty$. This proves Eq. (10). If reference actions are resampled, B here is the expected bounded correction under their fixed sampling law, as in the proof of Proposition 3. This expectation is taken after soft clipping when soft clipping is used.

The reference subtraction also removes a shared additive offset. Write u for the candidate successor value and $u_1, \ldots, u_N$ for the reference values. Translation equivariance of the empirical quantile gives

$$(u + c) - \text{Quantile}_\kappa(\{u_i + c\}_{i=1}^N) = u - \text{Quantile}_\kappa(\{u_i\}_{i=1}^N).$$

This identity concerns a common offset only; errors that differ across candidate and reference actions need not cancel. Neither identity implies policy invariance or a policy-improvement guarantee for the original reward.

C Experimental Details

C.1 Quantile Advantage Function Characterization and Regularization

To provide an overview of the learned quantile advantage function, we report summary statistics computed across 20 D4RL tasks using the best-performing hyperparameter setting (Appendix C.4). For each task, we aggregate quantile advantage values from four independent training runs and report: (1) the mean and standard deviation of both positive and negative quantile advantages, and (2) the proportion of samples exhibiting positive quantile advantages.

Table 2 describes the scale and sign of the learned signal across tasks. The positive fraction is the frequency with which a candidate's estimated successor value exceeds its behavior reference. It depends on the sampled candidates, κ, and the learned transition and value functions; it is not a measured probability of true policy improvement. These summaries characterize the correction used during learning, while the with/without comparison in Figure 5 evaluates its contribution to policy returns.

Table 2: Quantile advantage statistics for 20 D4RL tasks.

Task Name	**Positive**	**Negative**	**Pos. (%)**
halfcheetah-medium	$1.62_{\pm 0.2}$	$-0.11_{\pm 0.3}$	32.8
halfcheetah-medium-replay	$1.84_{\pm 0.3}$	$-0.90_{\pm 0.0}$	30.4
halfcheetah-medium-expert	$2.00_{\pm 0.2}$	$-1.75_{\pm 0.3}$	38.7
hopper-medium	$0.48_{\pm 0.1}$	$-0.14_{\pm 0.0}$	22.1
hopper-medium-replay	$1.25_{\pm 0.2}$	$-0.83_{\pm 0.1}$	38.3
hopper-medium-expert	$0.39_{\pm 0.2}$	$-0.55_{\pm 0.0}$	10.1
walker2d-medium	$0.70_{\pm 0.1}$	$-0.06_{\pm 0.0}$	43.5
walker2d-medium-replay	$1.87_{\pm 0.3}$	$-0.13_{\pm 0.0}$	20.0
walker2d-medium-expert	$2.33_{\pm 0.3}$	$-2.20_{\pm 0.5}$	26.2
antmaze-umaze	$1.70_{\pm 0.2}$	$-1.05_{\pm 0.1}$	52.8
antmaze-umaze-diverse	$0.03_{\pm 0.0}$	$-0.04_{\pm 0.0}$	34.0
antmaze-medium-play	$0.58_{\pm 0.1}$	$-0.40_{\pm 0.1}$	44.7
antmaze-medium-diverse	$0.48_{\pm 0.05}$	$-0.26_{\pm 0.05}$	54.0
antmaze-large-play	$0.58_{\pm 0.02}$	$-0.28_{\pm 0.05}$	48.3
antmaze-large-diverse	$0.42_{\pm 0.08}$	$-0.23_{\pm 0.02}$	62.3
pen-human	$2.34_{\pm 1.1}$	$-1.74_{\pm 0.6}$	41.3
pen-cloned	$1.14_{\pm 0.2}$	$-1.01_{\pm 0.1}$	33.4
kitchen-complete	$1.15_{\pm 0.2}$	$-0.85_{\pm 0.1}$	42.3
kitchen-partial	$0.47_{\pm 0.1}$	$-0.44_{\pm 0.2}$	31.1
kitchen-mixed	$0.52_{\pm 0.0}$	$-0.74_{\pm 0.0}$	33.8

Quantile Advantage Soft Clipping. We bound the correction applied to critic targets by soft clipping each computed quantile advantage:

$$\text{softclip}(x) = \begin{cases} \lambda_p \cdot \tanh\left(\frac{x}{\lambda_p}\right), & x \geq 0, \\ \lambda_n \cdot \tanh\left(\frac{x}{\lambda_n}\right), & x < 0, \end{cases}$$

where $\lambda_p, \lambda_n > 0$ set the positive and negative saturation levels, respectively. The implementation uses this bounded correction in place of λA in Eq. (5).

The transformation preserves the sign and ordering of its input, is approximately linear near zero, and compresses differences in the tails. Unlike hard clipping, it has a smooth change in slope. It bounds the correction magnitude by $\max(\lambda_p, \lambda_n)$; the corresponding fixed-point perturbation bound is given in the proof of Proposition 4.

A representative setting in the reported implementation is $\lambda_p = 6$ and $\lambda_n = 4$, with ratio 1.5. The two branches can therefore saturate at different magnitudes.

C.2 Diffusion Actor and Residual-Critic Architecture Design

This section describes the diffusion actor and the value-network architectures. The auxiliary transition and behavior models are described in Appendix C.5.

Diffusion Actor. The actor is instantiated as a denoising diffusion probabilistic model (DDPM) with a variance-preserving (VP) noise schedule. The noise predictor is a five-layer multilayer perceptron (MLP) with Mish activations. We use 10 denoising steps across all tasks.

Critic Networks. The value networks use either a standard MLP or a residual architecture comprising 16 residual blocks. Figure 8 shows the residual block, and Table 3 lists the critic backbone used for each task.

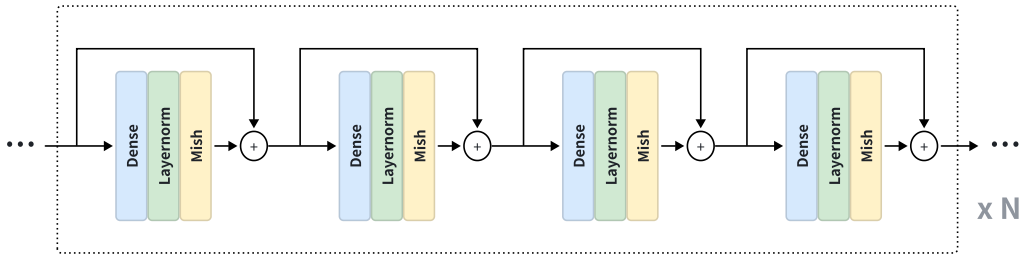
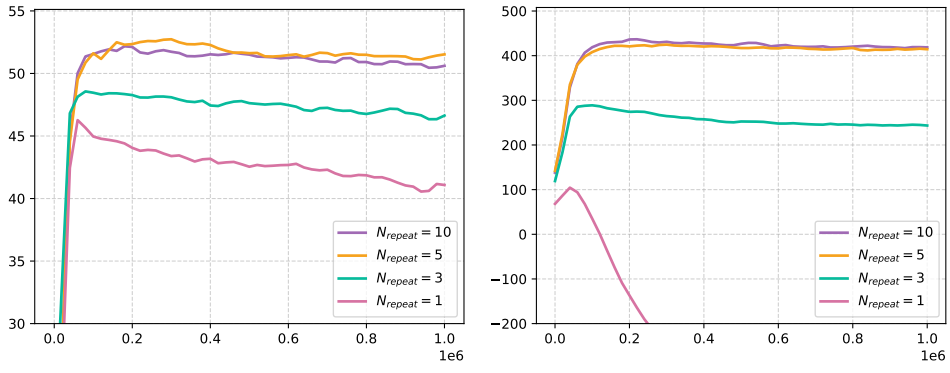


Figure 8: Residual block architecture used in critic networks.

C.3 Repeated Sampling for Value-Guided Diffusion Policies

The diffusion policy can provide multiple candidate actions at a state. We use repeated sampling in two different places: selecting actions for critic targets during training and selecting actions for environment interaction during evaluation. The candidate count controls the size of the set evaluated by the critic.

Training-Time Sampling: Max-Q Backup. For tasks using Max-Q backup (Kumar et al., 2020), critic targets are constructed from multiple next-action candidates. Table 3 gives the reported backup counts. These candidate-selection updates are practical control updates; Propositions 3–4 analyze the fixed-policy operator in Eq. (5). Figure 9 reports episode returns and mean predicted Q-values under different backup counts on `halfcheetah-medium-replay`.



(a) Episode returns under varying backup counts. (b) Training-Time average Q value per batch.

Figure 9: Returns and mean Q-values on `halfcheetah-medium-replay` with different backup counts.

Evaluation-Time Sampling: Q-Guided Inference. At evaluation time, we sample candidate actions from the diffusion model (typically 50–200 in the reported implementation). We transform their Q-values into softmax weights and draw the executed action from the resulting categorical distribution. This procedure favors candidates with larger estimated Q-values while retaining stochastic selection.

The backup count controls the critic target during training, whereas the evaluation candidate count controls action selection during rollout. Figure 9 varies the training-time backup count.

Table 3: Task-specific reference quantiles, Max-Q backup settings, backup counts n, and critic architectures.

Task	κ	Max Q Backup	n	Critic Net
halfcheetah-medium	0.75	True	5	ResNet
halfcheetah-medium-replay	0.75	True	5	ResNet
halfcheetah-medium-expert	0.75	True	10	ResNet
hopper-medium	0.75	False	1	ResNet
hopper-medium-replay	0.75	True	5	ResNet
hopper-medium-expert	0.95	False	1	ResNet
walker2d-medium	0.65	True	3	ResNet
walker2d-medium-replay	0.85	False	1	ResNet
walker2d-medium-expert	0.75	False	1	MLP
antmaze-umaze	0.55	True	10	ResNet
antmaze-umaze-diverse	0.65	True	10	ResNet
antmaze-medium-play	0.65	True	10	ResNet
antmaze-medium-diverse	0.65	True	10	ResNet
antmaze-large-play	0.65	True	10	ResNet
antmaze-large-diverse	0.55	True	10	ResNet
pen-human	0.65	True	3	MLP
pen-cloned	0.65	True	3	MLP
kitchen-complete	0.65	False	1	MLP
kitchen-partial	0.65	False	1	MLP
kitchen-mixed	0.65	False	1	MLP

C.4 Hyperparameter Setup

Table 3 reports the task-specific reference quantile, Max-Q backup setting, backup count, and critic backbone. Other parameters follow the DQL configuration (Wang et al., 2022).

C.5 Auxiliary Model Pretraining

The transition model predicts a successor from a state–action pair, and the DDPM behavior model supplies reference actions conditioned on the state. Both are pretrained on offline data. The transition network and DDPM noise predictor use MLPs with 256 units per layer and Mish activations. Table 4 lists the architectures and training settings: a 95%/5% train/test split, batch size 256, and 300 000 gradient steps with AdamW at learning rate 3×10^{-4}.

Each model trains in under 10 minutes on an NVIDIA RTX 4090. This timing covers auxiliary pretraining, excluding the subsequent actor–critic training run. The pretrained transition and behavior models can be reused when the dataset and their training configuration remain unchanged.

C.6 Dataset Construction for Offline RL in PointMaze

We use `PointMaze` from Gymnasium Robotics, a refactored version of D4RL's `Maze2D`. The agent navigates a closed maze using bounded 2D forces applied at 10 Hz. We omit goal-related fields from the observation, leaving a 4-dimensional vector of position and velocity. The reward is 1 upon

Table 4: Hyperparameters for Transition and Behavior Cloning Model Pretraining

Hyperparameter	Transition Model	Behavior Cloning Model
Architecture	4-layer MLP 256 neurons per layer	5-layer MLP 256 neurons per layer
Activation Function	Mish	Mish
Optimizer	AdamW	AdamW
Learning Rate	3×10^{-4}	3×10^{-4}
Batch Size	256	256
Gradient Descent Steps	300 000	300 000
Train/Test Split	95% / 5%	95% / 5%

reaching the fixed goal and 0 otherwise. In Figure 4, the green circle marks the start, the red star the goal, and beige squares the obstacles.

We collect trajectories during 10 independent runs of online SAC (Haarnoja et al., 2018), each lasting up to 250 000 training steps. The staged sampling schedule in Algorithm 2 periodically rolls out the current policy, capturing behavior at different stages of training.

We selected 853 successful trajectories, each shorter than 1000 steps, to form the offline dataset. Their routes are longer than the approximately 142-step reference route described in the original setup. The length bins in Table 5 contain 0.0%, 3.4%, 15.7%, and 80.9% of dataset trajectories, respectively. Figure 4 visualizes three route patterns: Left, Middle, and Right. The dataset contains 391 391 tuples $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}, r, \text{done})$.

Algorithm 2 Trajectory Sampling for Offline Dataset Construction

Initialize: policy π, environment $\mathcal{E}$, replay buffer $\mathcal{B}$
Define: a staged sampling schedule alternating coarse- and fine-grained rollouts
for each training step **do**
  Interact with $\mathcal{E}$ using π and store transitions $(\boldsymbol{s_t}, \boldsymbol{a_t}, r_t, \boldsymbol{s_{t+1}})$ into $\mathcal{B}$
  Periodically update π using mini-batches sampled from $\mathcal{B}$
  if sampling interval is triggered **then**
    Execute coarse-grained or fine-grained trajectory rollout according to schedule
    Store resulting trajectories in $\mathcal{D}$
  end if
end for
Filter suboptimal trajectories based on return and length heuristics
Return: offline dataset $\mathcal{D}_{\text{offline}}$

Table 5 uses disjoint length bins; its category names are descriptive labels, not certificates of global optimality. In particular, 15.7% denotes lengths in $[225, 300)$, whereas the fraction below 300 steps is 19.1% after also including the 3.4% in $[175, 225)$. This explains the cumulative < 300 statistic in Figure 4. The largest bin, 80.9%, contains trajectories of at least 300 steps.

Table 5: Distribution of trajectory quality based on length in the collected offline dataset.

Category	Length Range	Proportion
Optimal	< 175	0.0%
Near-Optimal	$[175, 225)$	3.4%
Competitive	$[225, 300)$	15.7%
Sub-Optimal	≥ 300	80.9%

D DIFFUSION BACKGROUND

Diffusion Model. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) consist of a forward process that corrupts data with noise and a reverse process that reconstructs data from noise. Specifically, the forward process is conducted by gradually adding Gaussian noise to samples $\boldsymbol{x_0}$ from an unknown data distribution $p_{\text{data}}(\boldsymbol{x_0})$, formulated as:

$$\begin{aligned} q(\boldsymbol{x_{1:T}} \mid \boldsymbol{x_0}) &:= \prod_{t=1}^{T} q(\boldsymbol{x_t} \mid \boldsymbol{x_{t-1}}), \\ q(\boldsymbol{x_t} \mid \boldsymbol{x_{t-1}}) &:= \mathcal{N}(\boldsymbol{x_t}; \sqrt{1-\beta_t}\boldsymbol{x_{t-1}}, \beta_t \boldsymbol{I}), \end{aligned} \tag{16}$$

where T denotes the total number of diffusion steps, and β_t controls the variance of the added noise at each step t. The reverse process is modeled as $p_\theta(\boldsymbol{x_{0:T}}) := p(\boldsymbol{x_T}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}} \mid \boldsymbol{x_t})$, and is trained by maximizing the evidence lower bound (ELBO) (Blei et al., 2017): $\mathbb{E}_q \left[\log \frac{p_\theta(\boldsymbol{x_{0:T}})}{q(\boldsymbol{x_{1:T}} \mid \boldsymbol{x_0})} \right]$. After training, samples can be generated by first drawing $\boldsymbol{x_T} \sim p(\boldsymbol{x_T})$ and then sequentially applying the learned reverse transitions to obtain $\boldsymbol{x_0}$. For conditional generation tasks, the reverse process can be extended to model $p_\theta(\boldsymbol{x_{t-1}} \mid \boldsymbol{x_t}, c)$, where c denotes the conditioning information.