Q-GUIDED DRIFTING POLICY: VALUE-GUIDED TARGET CONSTRUCTION FOR ONE-STEP OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning (RL) often requires expressive stochastic policies to represent heterogeneous and multimodal behavior distributions. Recently proposed drifting models provide a promising generative modeling framework for offline RL policies, enabling multimodal action distributions with one-step stochastic generation. However, offline RL only provides behavior actions rather than the high-value target actions required to train drifting policies, creating a mismatch between drifting policy learning and offline supervision. To bridge this gap, we propose **Q-Guided Drifting Policy(QGD)**, which uses a learned Q-function to construct training targets for drifting actors rather than directly optimizing the actor through critic gradients. QGD performs zeroth-order local refinement around multiple behavior samples, retaining high-value refined actions as positive targets and low-value behavior actions as additional negatives. The resulting policy preserves multiple high-value action modes while requiring only a single stochastic forward pass at inference. Experiments on OGBench and D4RL show that QGD achieves at least 95% of the best reported performance on 23 out of 37 tasks while retaining one-step stochastic inference.

1 INTRODUCTION

Offline reinforcement learning (RL) learns policies from fixed datasets that often contain heterogeneous behaviors and multiple feasible action modes (Fujimoto et al., 2019; Kumar et al., 2019; Yu et al., 2020; Levine et al., 2020; Rafailov et al., 2024; Mandlekar et al., 2022). This motivates the use of expressive generative policies in offline RL to represent complex and multimodal action distributions (Kang et al., 2023; Li et al., 2023; Zheng et al., 2023; Chen et al., 2023; Park et al., 2025b). Prior approaches improve such policies either by directly incorporating critic information into generative policy optimization, as in diffusion Q-learning (Wang et al., 2023), by using advantage- or value-weighted policy fitting (Peng et al., 2019; Kostrikov et al., 2022; Wang et al., 2020; Nair et al., 2021; Xu et al., 2023), or by using sampling-based policy extraction (Chen et al., 2023; Hansen-Estruch et al., 2023).

Many generative actors used in offline RL also face inference-efficiency challenges (Ding & Jin, 2024; Nguyen & Yoo, 2026; Wang et al., 2026), as diffusion policies typically require iterative denoising (Wang et al., 2023), while many flow-based policies rely on numerical integration at inference (Park et al., 2025b). drifting models (Deng et al., 2026) are particularly appealing in this context because they retain expressive stochastic distribution modeling while producing samples with a single forward pass at inference. This makes drifting a promising policy parameterization for offline RL. Recent works investigate the application of drifting models in RL, such as drifting Field Policy (DFP) (Koo et al., 2026), adapting drifting models to policy improvement through critic-guided action selection.

However, adapting drifting models as policies in offline RL introduces a gap between the target samples required by the drifting objective and the learning signals available in offline RL. Drifting models learn from samples of a target distribution, whereas offline RL provides behavior actions together with learned values, rather than samples from an improved policy distribution. A natural adaptation would be to incorporate critic gradients into the drifting policy learning, but directly

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.

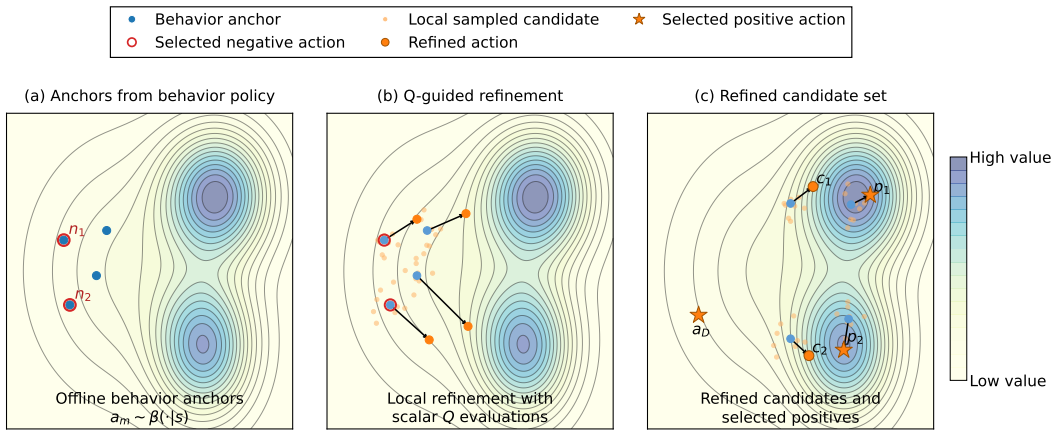

Figure 1: **Behavior-anchored Q-guided candidate refinement in QGD.** Candidate search starts from multiple offline behavior anchors, and detached scalar Q evaluations guide independent local refinement around each anchor to construct candidates for subsequent target construction.

applying critic gradients would alter the balance between the attraction and self-repulsion terms (Deng et al., 2026; Turan et al., 2026; Brown et al., 2026; Cao et al., 2026). This raises a central question: how can the fixed data distribution available in offline RL be transformed into suitable target-distribution samples for drifting policies?

In this work, we propose **Q-Guided Drifting (QGD)**, a practical framework that enables drifting policy learning in offline RL. To our knowledge, QGD is the first to explicitly construct behavior-supported, multimodal target distributions for critic-guided drifting learning. Instead of relying on critic gradients for policy optimization or critic-based candidate selection at inference, QGD uses the learned critic to construct the target actions for drifting policy learning. QGD has two key design components: i) Q-guided local refinement constructs target samples around behavior actions while preserving multiple high-value modes; and ii) low-value behavior negatives further steer the drifting policy away from undesirable action regions. QGD independently refines multiple behavior-model actions through local Q-guided search and uses the resulting actions and selected low-value behavior actions as positive and negative samples for drifting policy learning. Figure 1 illustrates this target-construction process.

QGD offers three desirable properties for offline generative policy learning. First, by keeping critic gradients outside the drifting field, QGD preserves the attraction–repulsion balance underlying the drifting equilibrium (Deng et al., 2026; Turan et al., 2026). Second, maintaining multiple independent local proposals allows the refinement process to preserve distinct high-value modes. Our theoretical analysis characterizes this multimodal preservation property of QGD during refinement, showing that the proposed refinement can retain diverse high-value solutions. Third, since the additional refinement is used only during training, QGD retains a stochastic one-step actor at deployment.

In the OGBench and D4RL benchmarks (Park et al., 2025a; Fu et al., 2020), QGD achieves at least 95% of the best reported performance on 23 out of 37 tasks. Our ablations further show that preserving multiple target actions consistently outperforms single-target compression, while per-anchor refinement improves the efficiency of constructing high-value candidates. These results validate the main design choices of QGD and demonstrate that value-guided policy improvement can be effectively combined with expressive one-step generative policy learning.

2 Related Work

Offline Reinforcement Learning with Generative Models. Generative models have been widely used for policy modeling and decision making (Janner et al., 2022; Chi et al., 2023), and expressive generative policies have been increasingly studied in offline RL for behavior modeling and policy improvement (Kang et al., 2023; Wang et al., 2023; Chen et al., 2023; Park et al., 2025b). Across these methods, learned value information is incorporated through policy optimization (Peng et al.,

2019; Kang et al., 2023; Wang et al., 2023; Park et al., 2025b; Fang et al., 2025), regularization (Chen et al., 2024a;c; Zhang et al., 2024; 2026), or generative guidance (Psenka et al., 2024; Chen et al., 2024b; Lu et al., 2023; Mao et al., 2024; Zhang et al., 2025; Li & Levine, 2026; Xu et al., 2026). A complementary line of work improves the computational efficiency of expressive generative policies through consistency-based generation (Ding & Jin, 2024), shortcut models (Frans et al., 2025), and native one-step generative parameterizations (Nguyen & Yoo, 2026; Wang et al., 2026). Collectively, these approaches use value information to directly improve the generative policy itself by guiding the generation process toward high-value actions. In contrast, QGD separates value-based improvement from generative policy learning by first constructing a value-guided target action distribution and then training drifting policies on the resulting samples.

Sampling-Based Policy Extraction. Sampling-based approaches improve policies by generating candidate actions and leveraging scalar evaluations for optimization or extraction. CEM (Rubinstein, 1999) and MPPI (Williams et al., 2017) iteratively refine sampling distributions using scalar objectives. In offline RL, SfBC (Chen et al., 2023) and IDQL (Hansen-Estruch et al., 2023) share this sampling-and-evaluation structure, using generative behavior models to propose actions and learned critics for policy extraction. QGD locally refines multiple behavior samples using scalar Q evaluations, preserving distinct high-value modes as distributional targets for policy learning.

Drifting Models. Drifting models (Deng et al., 2026) introduce a one-step generative framework that updates the pushforward distribution during training toward a target distribution, avoiding iterative sampling at inference. Subsequent work has studied drifting through score matching (Turan et al., 2026), Wasserstein gradient flows (Cao et al., 2026), generalized kernel dynamics (Esteban-Casadevall et al., 2026), and second-order formulations (Brown et al., 2026).

3 PRELIMINARIES

Offline Reinforcement Learning. We consider an offline reinforcement learning problem defined by a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s'|s,a)$ is the transition dynamics, $r(s,a)$ is the reward function, and $\gamma \in (0,1)$ is the discount factor. The agent is given a fixed dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{N_\mathcal{D}}$ collected by an unknown behavior policy π_β, without any additional interaction with the environment during training (Levine et al., 2020). The goal of offline RL is to improve the policy while limiting its deviation from the behavior policy that generated the dataset. A standard formulation is (Wu et al., 2019)

$$\pi^* = \arg\max_{\pi \in \Pi} \quad \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \qquad \text{s.t.} \quad \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}\big(\pi(\cdot|s) \,\|\, \pi_\beta(\cdot|s)\big)\right] \le \epsilon, \tag{1}$$

where ϵ controls how far the learned policy is allowed to deviate from the behavior policy.

Drifting Models. Drifting models formulate one-step generation as the training-time evolution of a pushforward distribution (Deng et al., 2026). Let $\epsilon \sim p_\epsilon$ and $x = f_\theta(\epsilon)$, which induces the model distribution $q_\theta = (f_\theta)_\# p_\epsilon$. Given a target distribution p and the current model distribution q_θ, drifting defines the field

$$V_{p,q_\theta}(x) = \frac{\mathbb{E}_{y^+\sim p}\left[k(x,y^+)(y^+ - x)\right]}{\mathbb{E}_{y^+\sim p}\left[k(x,y^+)\right]} - \frac{\mathbb{E}_{y^-\sim q_\theta}\left[k(x,y^-)(y^- - x)\right]}{\mathbb{E}_{y^-\sim q_\theta}\left[k(x,y^-)\right]}, \tag{2}$$

where $k(\cdot,\cdot)$ is a positive kernel. The first term attracts generated samples toward the target distribution, while the second provides a self-repulsive correction from the current model distribution. When $p = q_\theta$, the two terms cancel and $V_{p,q_\theta}(x) = 0$. Drifting trains the one-step generator by regressing each sample toward its drifted target with stop-gradient:

$$\mathcal{L}_{\mathrm{drift}}(\theta) = \mathbb{E}_{\epsilon\sim p_\epsilon}\left[\left\| f_\theta(\epsilon) - \mathrm{sg}\left(f_\theta(\epsilon) + V_{p,q_\theta}\left(f_\theta(\epsilon)\right)\right)\right\|_2^2\right], \tag{3}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. Thus, the drifting objective only requires samples from the target distribution p to construct the attraction field; the mechanism used to specify or construct p is separate from the regression objective.

Candidate Evaluation with Scalar Q. A common strategy in offline RL and planning is to generate a finite set of candidate actions and evaluate them using a learned critic (Chen et al., 2023; Hansen-Estruch et al., 2023; Argenson & Dulac-Arnold, 2020; Sikchi et al., 2022). Given a state s, candidate

actions can be sampled from an action distribution μ as

$$a^1, \ldots, a^N \sim \mu(\cdot|s). \tag{4}$$

The candidates are evaluated by their scalar action values $Q(s, a^1), \ldots, Q(s, a^N)$. A standard best-of-$N$ selection rule chooses $a^\star = \arg\max_{a^i} Q(s, a^i)$.

4 Q-Guided Drifting Policy

QGD separates target policy construction from one-step policy learning. For each state s, we first use the offline behavior distribution and the learned critic to construct a target action distribution $p_{\text{tar}}(a|s)$, and then train a drifting policy $\pi_\theta(a|s)$ to match this distribution. This decomposition is particularly suitable for drifting, whose training objective only requires samples from a target distribution. All candidate generation and critic evaluation are therefore restricted to training, while the learned drifting policy generates actions with a single stochastic forward pass at inference.

To specify the target distribution, we start from the standard KL-regularized policy improvement objective

$$\pi^*(\cdot|s) = \arg\max_{\pi} \left\{ \mathbb{E}_{a\sim\pi(\cdot|s)} \left[Q_\phi(s, a) \right] - \alpha D_{\text{KL}} \left(\pi(\cdot|s) \| \pi_\beta(\cdot|s) \right) \right\}, \tag{5}$$

where π_β denotes the behavior policy and $\alpha > 0$ is the regularization coefficient. For a fixed critic, the optimal solution has the Boltzmann form

$$\pi^*(a|s) = \frac{1}{Z(s)} \pi_\beta(a|s) \exp\left(\frac{Q_\phi(s, a)}{\alpha} \right), \tag{6}$$

where $Z(s)$ is the normalization constant. Thus, QGD uses this behavior-regularized Boltzmann policy as the target distribution and constructs a finite approximation of it from behavior-conditioned candidate actions and scalar critic evaluations.

4.1 Q-Guided Local Refinement

To construct the target distribution introduced above, QGD starts from samples of a learned behavior policy and performs Q-guided refinement around multiple behavior-conditioned anchors. Rather than fitting a single proposal to all candidate actions, we maintain one local proposal for each anchor and update these proposals independently. This allows the refinement procedure to retain candidate actions from different regions of the behavior distribution.

For each state s, we first draw M actions from the learned behavior proposal,

$$a_m = \text{clip}\left(\beta(s, \xi_m), -1, 1 \right), \qquad \xi_m \sim \mathcal{N}(0, I), \qquad m = 1, \ldots, M. \tag{7}$$

We associate each behavior action a_m with an independent diagonal Gaussian proposal,

$$q_m^0(a|s) = \mathcal{N}\left(a; \mu_m^0, \text{diag}\big((\sigma_m^0)^2\big) \right), \qquad \mu_m^0 = a_m, \qquad \sigma_m^0 = \sigma_0. \tag{8}$$

This gives M local proposal distributions initialized from different samples of the behavior policy. The remaining design choice is how these proposals are updated during refinement. A straightforward alternative is to fit a single Gaussian sampling distribution to all candidates and iteratively refit it toward high-value samples. In contrast, QGD associates each behavior sample with its own Gaussian sampling distribution and refines these local distributions independently. This allows multiple high-value regions to remain represented throughout refinement, rather than being merged into a single Gaussian proposal.

At refinement step t, the m-th local proposal samples N candidate actions,

$$a_{m,n}^t = \text{clip}\left(\mu_m^t + \widetilde{\sigma}_m^t \odot \epsilon_{m,n}^t, -1, 1 \right), \qquad \epsilon_{m,n}^t \sim \mathcal{N}(0, I), \qquad n = 1, \ldots, N. \tag{9}$$

where $\widetilde{\sigma}_m^t$ denotes the stabilized proposal scale used for sampling.

We evaluate each candidate using the critic, $q_{m,n}^t = \text{sg}\left[\overline{Q}_\phi\left(s, a_{m,n}^t \right) \right]$. The candidates are assigned normalized exponential weights,

$$\omega_{m,n}^t = \frac{\exp\big(\hat{q}_{m,n}^t / \tau_{\text{refine}}\big)}{\sum_{u=1}^{N} \exp\big(\hat{q}_{m,u}^t / \tau_{\text{refine}}\big)}, \qquad \hat{q}_{m,n}^t = \frac{q_{m,n}^t - \text{mean}_u\, q_{m,u}^t}{\text{std}_u\, q_{m,u}^t + \epsilon_Q}. \tag{10}$$

where $\tau_{\text{refine}} > 0$ is a temperature parameter. We then refit each local Gaussian proposal using the weighted empirical moments of its local samples,

$$\mu_m^{t+1} = \sum_{n=1}^{N} \omega_{m,n}^t a_{m,n}^t, \qquad \left(\sigma_m^{t+1}\right)^2 = \sum_{n=1}^{N} \omega_{m,n}^t \left(a_{m,n}^t - \mu_m^{t+1}\right)^2. \tag{11}$$

This update is repeated for T_r refinement steps. Crucially, the Gaussian parameters are updated separately for each m: candidates associated with different behavior samples are never pooled when estimating μ_m^t and σ_m^t. Unlike a single Gaussian proposal that summarizes all candidates with one mean and variance, this local refinement maintains multiple independently updated proposals, allowing multiple high-value regions to remain represented throughout refinement.

After T_r refinement steps, each local proposal yields a refined candidate $c_{m,s}$. We collect the independently refined candidates as $\mathcal{C}_s = \{c_{m,s}\}_{m=1}^M$. These refined candidates are subsequently used to construct the positive samples for drifting. The following proposition characterizes the distribution approximated by the local refinement.

Proposition 1. *For a fixed state s, let $p_{\text{QGD}}(a|s)$ denote the target distribution obtained by applying the proposed Q-guided local refinement. Let $\pi^*(a|s) = \frac{1}{Z(s)}\pi_\beta(a|s)\exp(Q_\phi(s,a)/\alpha)$ be the Boltzmann-refined behavior distribution. Assume that the finite-step local refinement incurs at most ϵ_{ref} total-variation error relative to the corresponding ideal local Boltzmann refinements, and that the local Gaussian mixture approximation satisfies $D_{\text{TV}}(\frac{1}{M}\sum_m q_m^0(\cdot|s), \pi_\beta(\cdot|s)) \le \epsilon_\beta$, and the local normalization constants satisfy $|Z_m(s) - Z(s)| \le \epsilon_Z$, $Z_m(s), Z(s) \ge \delta$. Then,*

$$D_{\text{TV}}(p_{\text{QGD}}(\cdot|s), \pi^*(\cdot|s)) \le \epsilon_{\text{ref}} + \epsilon_\beta + C\frac{\epsilon_Z}{\delta^2},$$

where $C = \frac{1}{2}\sup_a \exp(Q_\phi(s,a)/\alpha)$ is the maximum scaling factor induced by the Boltzmann weighting.

Proposition 1 shows that the proposed Q-guided local refinement constructs a target distribution that is boundedly close to the Boltzmann-refined behavior distribution. By performing refinement around multiple behavior-conditioned local proposals, QGD preserves multiple high-value action regions rather than collapsing the target distribution into a single mode. These refined targets are then used as supervision for the drifting update. The proof is provided in Appendix B.

4.2 Q-Guided Drifting Policy Learning

The local refinement in Section 4.1 produces candidate actions around multiple behavior samples. However, finite refinement does not guarantee that every local proposal reaches a high-value region. Treating all refined candidates as positive samples can therefore introduce low-value actions into the drifting target. We first exclude the refined candidate with the highest critic value from positive selection and then retain the K_+ highest-valued candidates among the remainder.

$$\mathcal{P}_s = \{a_{\mathcal{D}}\} \cup \text{TopK}_{K_+}\left(\mathcal{C}_s \setminus \{c_s^\star\}; \overline{Q}_\phi(s,\cdot)\right), \qquad c_s^\star = \arg\max_{c\in\mathcal{C}_s} \overline{Q}_\phi(s,c). \tag{12}$$

The selected set $\mathcal{P}_s$ serves as the positive samples for drifting.

Drifting additionally uses negative samples to define the repulsive component of its update. Standard drifting reuses samples from the current model distribution as negatives. We augment these samples with low-value actions from the initial behavior proposal. Specifically, among the behavior samples $\{a_m\}_{m=1}^M$ used to initialize local refinement, we select

$$\mathcal{N}_s = \text{BottomK}_{K_-}\left(\{a_m\}_{m=1}^M; \overline{Q}_\phi(s,\cdot)\right). \tag{13}$$

This negative-sample augmentation admits the same interpretation as the training-time guidance mechanism introduced for drifting. Let $q_\theta(\cdot|s)$ denote the current policy distribution and let $q_{\bar{\beta}}^-(\cdot|s)$ denote the distribution induced by the low-value behavior samples in $\mathcal{N}_s$. Augmenting the standard model negatives with these samples corresponds to using the negative distribution

$$\widetilde{q}(\cdot|s) = (1-\lambda_-)q_\theta(\cdot|s) + \lambda_- q_{\bar{\beta}}^-(\cdot|s), \tag{14}$$

where $\lambda_- \in [0, 1)$ controls the contribution of the additional behavior negatives. If $p^+(\cdot|s)$ denotes the distribution represented by the selected positive samples, the equilibrium condition $\widetilde{q}(\cdot|s) = p^+(\cdot|s)$ implies

$$q_\theta(\cdot|s) = \kappa p^+(\cdot|s) - (\kappa - 1)q_\beta^-(\cdot|s), \qquad \kappa = \frac{1}{1-\lambda_-}. \tag{15}$$

Thus, the additional low-value behavior negatives induce an extrapolative guidance effect: the policy is attracted toward the high-value refined actions while being pushed away from low-value behavior actions. This mirrors the CFG construction in drifting, with critic-based action preference replacing class conditioning.

For policy learning, we use $\mathcal{P}_s$ as the positive sample set and $\mathcal{Y}_s^- = \mathcal{X}_s \cup \mathcal{N}_s$ as the negative sample set, where $\mathcal{X}_s$ denotes samples from the current policy. We train the actor with the standard stop-gradient drifting objective,

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\,\epsilon\sim\mathcal{N}(0,I)}\left[\left\| f_\theta(s,\epsilon) - \mathrm{sg}\left(f_\theta(s,\epsilon) + V_{\mathcal{P}_s,\mathcal{Y}_s^-}\left(f_\theta(s,\epsilon)\right)\right)\right\|_2^2\right]. \tag{16}$$

Additional training details and the complete algorithm are provided in Appendix D.1.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate QGD on the OGBench (Park et al., 2025a) and D4RL (Fu et al., 2020) benchmarks. OGBench provides challenging offline RL tasks with diverse and suboptimal behavior datasets, covering long-horizon navigation and multi-stage robotic manipulation. We consider five task suites, including Cube, Scene, and Puzzle environments, resulting in 25 state-based tasks. We further evaluate QGD on standard D4RL offline RL benchmarks, including 12 Adroit tasks that evaluate dexterous manipulation under diverse offline behavior distributions. Together, these constitute the 37 tasks used for our aggregate cross-method comparison. We additionally evaluate QGD in an offline-to-online setting; the corresponding protocol and results are provided in Appendix E. Detailed environment and dataset configurations are provided in Appendix C.

Baselines. For our offline RL experiments, we compare QGD against representative methods covering conventional offline RL policies and generative policy approaches. (1) *Conventional policies.* We include BC, IQL (Kostrikov et al., 2022), and ReBRAC (Tarasov et al., 2023) as representative actor-critic offline RL baselines. (2) *Flow-based generative policies.* We compare with FQL (Park et al., 2025b) and its flow-based variants FBRAC and IFQL, as well as recent value-guided flow-policy methods QAM and VGF. (3) *One-step generative policies.* We further compare with one-step generative actors, including FQL, Shortcut-FQL, and MeanFlow-FQL, which provide the closest comparison under one-step action generation. Detailed baseline descriptions are provided in Appendix D.4.

Evaluation Protocol. For state-based OGBench and D4RL tasks, all agents are trained for 10^6 gradient steps and evaluated every 10^5 steps using 50 episodes per task. For each random seed, we average the performance over the final three evaluation checkpoints at 8×10^5, 9×10^5, and 10^6 steps. We run each method with four random seeds and report the mean and standard deviation across seeds. Hyperparameters, network architectures, and additional training details are provided in Appendix D.

5.2 Main Offline RL Results

Figure 2 summarizes the offline RL performance of QGD and the baselines on OGBench and D4RL. Since the two benchmarks use different evaluation metrics, we report their aggregate results separately and compute averages over the tasks jointly reported by all compared methods.

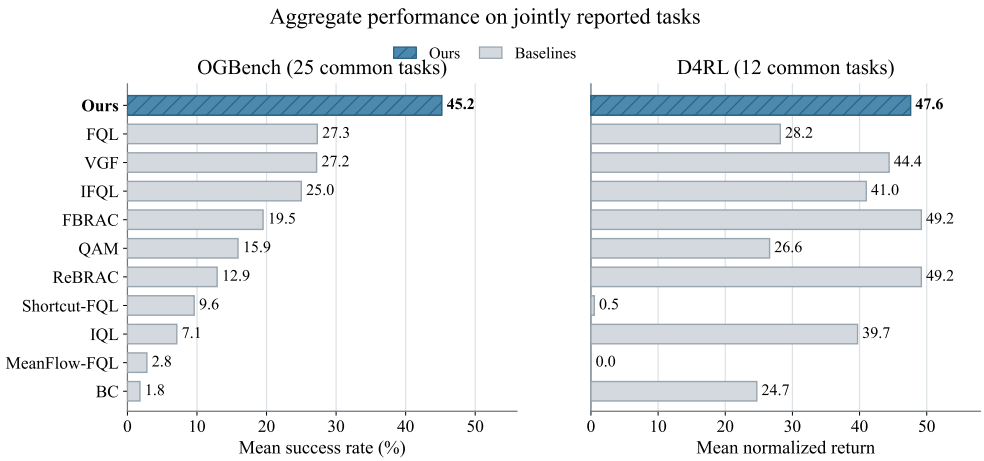

Figure 2: **Aggregate performance on OGBench and D4RL benchmarks.** Bars show the unweighted mean of task-level scores from Table 5. OGBench and D4RL are summarized separately because they report success rate and normalized return, respectively. Each panel includes only tasks with results for every displayed method (25 OGBench tasks and 12 D4RL tasks). The complete per-task results are provided in the Appendix E.

On the 25 reported OGBench tasks, QGD achieves the best average performance of 45.2, substantially outperforming FQL (27.3), VGF (27.2), and the remaining baselines. This result shows that QGD is particularly effective on the challenging OGBench tasks, where policy improvement must be performed from diverse and suboptimal offline behavior.

On the 12 reported D4RL tasks, QGD achieves an average score of 47.6. While ReBRAC (49.2) and FBRAC (49.2) obtain higher averages on this subset, QGD remains competitive, outperforming VGF (44.4), IFQL (41.0), and the other evaluated baselines. Overall, these results demonstrate that QGD achieves strong performance across both benchmarks, with its clearest advantage on OGBench.

5.3 Ablation Study

We study the main design choices of QGD from two perspectives: the mechanisms used to construct the positive and negative samples for the drifting policy, and the sensitivity to the hyperparameters of the refinement procedure. Additional ablations are provided in Appendix F.

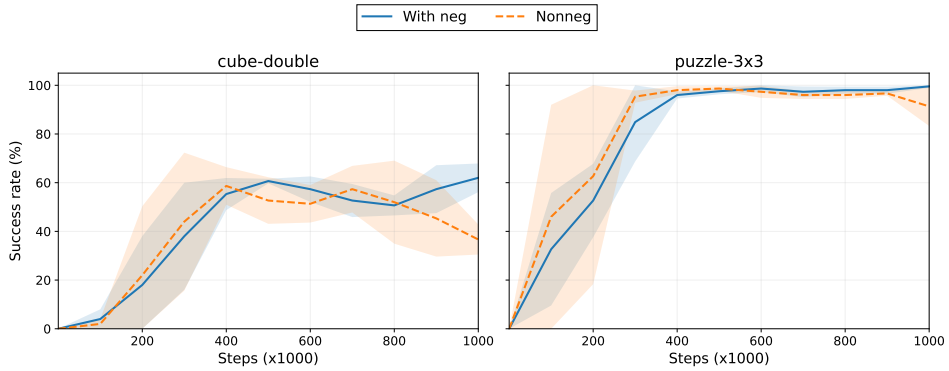

Figure 3: Effect of low-Q behavior actions as negative references.

Target-construction mechanisms. We first examine how the candidate set used for drifting policies should be constructed. Table 1 compares QGD with two groups of variants. The first group studies target aggregation, where *Single best* and *Advantage-weighted target* replace the multi-target representation with a single selected target or a weighted target. The second group studies the

Table 1: **Target-construction ablations.** Values are success rates (%) reported as mean ± standard deviation. AntMaze uses the offline-to-online setting. Macro averages weight the four environments equally.

Variant	AntMaze	Cube	Scene	Puzzle	Average
QGD	$84 \pm 2 \rightarrow 97 \pm 1$	46 ± 14	95 ± 7	92 ± 6	**83**
(a) Single-target compression					
Single best	$86 \pm 1 \rightarrow 98 \pm 1$	1 ± 2	53 ± 24	74 ± 15	57
Advantage-weighted target	$78 \pm 2 \rightarrow 95 \pm 6$	14 ± 2	84 ± 2	98 ± 1	73
(b) Refinement, selection, and weighting					
No refinement ($K = 8$)	$31 \pm 11 \rightarrow 90 \pm 4$	2 ± 2	30 ± 11	0 ± 0	31
Budget-matched raw proposals ($K = 272$)	$82 \pm 4 \rightarrow 99 \pm 1$	28 ± 5	94 ± 2	100 ± 0	80
Without dataset action (size preserved)	$81 \pm 4 \rightarrow 80 \pm 5$	14 ± 9	93 ± 7	88 ± 9	69
Global Top-K	$84 \pm 3 \rightarrow 99 \pm 1$	16 ± 6	99 ± 1	100 ± 0	79
Random-drop Top-K	$78 \pm 4 \rightarrow 98 \pm 2$	18 ± 3	98 ± 0	100 ± 0	79
Uniform positive weights	$74 \pm 3 \rightarrow 90 \pm 6$	31 ± 7	96 ± 2	100 ± 0	79

target-generation procedure, including the dataset-action inclusion, candidate selection, positive weighting, and local refinement.

Variant definitions. Single best retains the highest-valued selected target, while *Advantage-weighted target* replaces the selected targets with one value-weighted target. *No refinement* uses $K = 8$ raw proposals, and *Budget-matched raw proposals* uses $K = 272$ to match QGD's evaluation budget. *Without dataset action* removes $a_{\mathcal{D}}$ while preserving target-set size. *Global Top-K* selects the K_+ highest-valued refined candidates; *Random-drop Top-K* instead drops one candidate at random before selection. *Uniform positive weights* assigns unit weight to refined positives while retaining $w_{\mathcal{D}}$.

Compressing the selected candidates into a single target substantially degrades performance: *Single best* and *Advantage-weighted target* obtain averages of 57 and 73, respectively, compared with 83 for QGD. The degradation is particularly pronounced on Cube, where retaining multiple refined targets improves the success rate from 1 and 14 to 46. These results indicate that preserving multiple value-preferred actions is important for training a stochastic drifting policy.

The remaining variants further isolate the role of refinement and sample selection. The *No refinement* variant performs poorly under a limited candidate budget, while *Budget-matched raw proposals* recover competitive performance, indicating that local refinement mainly improves the efficiency of finding high-value actions. The *Global top-K* variant, which retains the highest-scoring refined candidate instead of excluding it, remains competitive but obtains a lower average score than QGD (79 versus 83), suggesting a modest benefit from the highest-value exclusion. Other variants that alter dataset-action inclusion or positive weighting also remain relatively strong, indicating that the main gain comes from retaining multiple high-value positive samples rather than from a specific selection or weighting rule.

Effect of low-value behavior negatives. We examine the effect of removing low-value behavior actions from the negative references in the drifting objective. This ablation is evaluated on `CubeDouble-task1` and `Puzzle3x3-task1`. We compare the standard setting with low-Q behavior actions as negative references against a variant that removes these negative samples.

As shown in Figure 3, removing low-Q behavior actions as negative references degrades performance on the evaluated tasks. This indicates that negative references provide additional distributional guidance beyond positive target construction alone, helping the learned policy reduce probability mass assigned to low-value behavior regions.

Sensitivity to refinement hyperparameters. We next study the sensitivity of QGD to the main parameters controlling the training-time search procedure. Figure 4 varies the number of behavior-conditioned anchors and the number of candidates refined around each anchor. Increasing either quantity provides more candidate coverage and critic evaluations, while performance generally improves when moving away from very small candidate populations. These results support the role of multi-local refinement: multiple anchors improve coverage of distinct behavior regions, while additional candidates strengthen the local Q-guided search around each anchor.

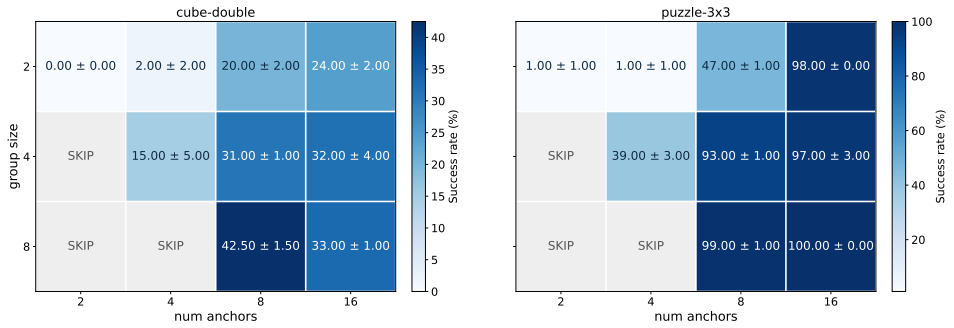

Figure 4: Sensitivity to the number of behavior anchors and candidates per anchor.

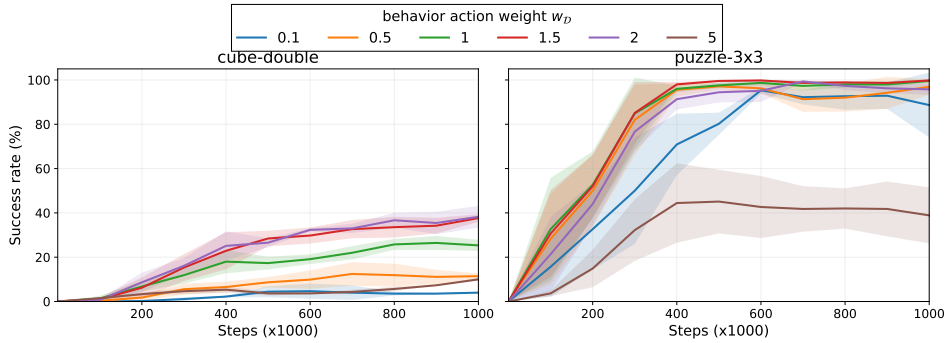

Figure 5: Sensitivity to the behavior-action weight $w_{\mathcal{D}}$.

We also examine the sensitivity of the behavior-action weighting used in drifting, with results shown in Figure 5. Across a range of behavior-action weights, QGD maintains competitive performance, while overly small or large weights can degrade the performance. This suggests that the behavior-action weight controls the trade-off between refined high-value targets and behavior regularization, and that QGD remains effective within a reasonable range of this trade-off.

6 Conclusion

We introduced QGD, a one-step generative policy for offline reinforcement learning that separates critic-guided action improvement from generative policy learning. Rather than incorporating critic gradients or an additional value-maximization objective into the actor, QGD uses detached scalar critic evaluations to refine behavior-regularized action candidates and construct positive and negative samples for drifting. The resulting supervision is then amortized into a stochastic one-step policy using the standard drifting objective, confining candidate search and critic evaluation entirely to training. Experiments on OGBench and D4RL demonstrate strong offline RL performance while retaining efficient one-step inference, and our ablations highlight the importance of preserving multiple high-value target actions rather than compressing critic information into a single action. These results suggest that expressive generative policies can benefit from using learned values to construct their training samples.

AI Use Statement

Generative AI tools were used during the preparation of this work to assist with manuscript drafting and editing, literature search and organization, and the refinement of mathematical exposition and experimental presentation. The tools were used only as assistive resources rather than autonomous research agents. All methodological decisions, mathematical statements, experimental results, citations, and scientific claims were reviewed and verified by the authors. The authors take full responsibility for the content of the paper.

Reproducibility Statement

We provide the complete specification of QGD in the main paper, including behavior-conditioned candidate generation, local refinement, positive and negative sample construction, and the resulting actor and critic objectives. The appendix provides additional implementation details, network architectures, hyperparameters, benchmark and dataset specifications, and the complete evaluation protocol, including training budgets, evaluation frequency, checkpoint aggregation, and random seeds. Additional derivations and proofs for the theoretical results are also provided in the appendix. The code repository is available at: `https://anonymous.4open.science/r/qgd-ACDB/README.md`

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations (ICLR)*, 2023.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *ArXiv*, abs/2008.05556, 2020.

Drake Brown, Yuhao Huang, Shih-Hsin Wang, and Bao Wang. Second order drifting models. *ArXiv*, abs/2608.07924, 2026.

Jiarui Cao, Zixuan Wei, and Yuxin Liu. Gradient flow drifting: Generative modeling via wasserstein gradient flows of kde-approximated divergences. *ArXiv*, abs/2603.10592, 2026.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. In *International Conference on Learning Representations (ICLR)*, 2024a.

Huayu Chen, Kaiwen Zheng, Hang Su, and Jun Zhu. Aligning diffusion behaviors with q-functions for efficient continuous control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. Diffusion policies creating a trust region for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024c.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *ArXiv*, abs/2602.04770, 2026.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Maria Esteban-Casadevall, Jorge Carrasco-Pollo, Max Welling, Jan-Willem van de Meent, Erik J. Bekkers, and Floor Eijkelboom. Kernel-gradient drifting models. *ArXiv*, abs/2605.10727, 2026.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bingyi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Daniele Foffano, Alessio Russo, and Alexandre Proutiere. Receding-horizon control via drifting models. *ArXiv*, abs/2604.04528, 2026.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

Yuxuan Gao, Yedong Shen, Shiqi Zhang, Wenhao Yu, Yifan Duan, Jia Pan, Jiajia Wu, Jiajun Deng, and Yanyong Zhang. Drift-based policy optimization: Native one-step policy learning for online robot control. *arXiv preprint arXiv:2604.03540*, 2026.

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*, abs/2304.10573, 2023.

Longxiang He, Li Shen, Linrui Zhang, Junbo Tan, and Xueqian Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *ArXiv*, abs/2310.05333, 2024.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

Jihwan Jeong, Xiaoyu Wang, Jingmin Wang, Scott Sanner, and Pascal Poupart. Reflect-then-plan: Offline model-based planning through a doubly Bayesian lens. In *International Conference on Machine Learning (ICML)*, 2025.

Zhou Jiang, Yandong Wen, and Zhen Liu. Drifting preference optimization for one-step generative models, 2026.

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 (NeurIPS)*, 2023.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Juil Koo, Mingue Park, Jiwon Choi, Yunhong Min, and Minhyuk Sung. Drifting field policy: A one-step generative policy via wasserstein gradient flow. *ArXiv*, abs/2605.07727, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

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 (NeurIPS)*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning (ICML)*, 2021.

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

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *International Conference on Learning Representations (ICLR)*, 2026.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning (ICML)*, 2023.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Haofei Lu, Yifei Shen, Dongsheng Li, Junliang Xing, and Dongqi Han. Habitizing diffusion planning for efficient and effective decision making. In *International Conference on Machine Learning (ICML)*, 2025.

Susie Lu, Haonan Chen, Weirui Ye, and Yilun Du. Driftworld: Fast world modeling through drifting. *ArXiv*, abs/2607.15065, 2026.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

Liyuan Mao, Haoran Xu, Xianyuan Zhan, Weinan Zhang, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *ArXiv*, abs/2412.06685, 2024.

Kazuki Mizuta and Karen Leung. Unified generation-refinement planning: Bridging guided flow matching and sampling-based mpc for social navigation. *ArXiv*, abs/2508.01192, 2025.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *ArXiv*, abs/2006.09359, 2021.

Xuan Thanh Nguyen and Chang Yoo. One-step flow q-learning: Addressing the diffusion policy bottleneck in offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025b.

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

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning (ICML)*, 2024.

Rafael Rafailov, Kyle Hatch, Anikait Singh, Laura Smith, Aviral Kumar, Ilya Kostrikov, Philippe Hansen-Estruch, Victor Kolev, Philip Ball, Jiajun Wu, Chelsea Finn, and Sergey Levine. D5RL: Diverse datasets for data-driven deep reinforcement learning. *ArXiv*, abs/2408.08441, 2024.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning (CoRL)*, 2022.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Erkan Turan, Nicolas Dufour, and Maks Ovsjanikov. Generative drifting is secretly score matching: a spectral and variational perspective. *ArXiv*, abs/2603.09936, 2026.

Siddarth Venkatraman, Shivesh Khaitan, Ravi Tej Akella, John Dolan, Jeff Schneider, and Glen Berseth. Reasoning with latent diffusion in offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Zeyuan Wang, Da Li, Yulin Chen, Ye Shi, Liang Bai, Tianyuan Yu, and Yanwei Fu. One-step generative policies with q-learning: A reformulation of meanflow. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S. Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40: 344–357, 2017.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations (ICLR)*, 2023.

Haoran Xu, Kaiwen Hu, Somayeh Sojoudi, and Amy Zhang. Reinforcement learning via value gradient flow. In *International Conference on Learning Representations (ICLR)*, 2026.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B. Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

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 *International Conference on Learning Representations (ICLR)*, 2026.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky T. Q. Chen. Guided flows for generative modeling and decision making. *ArXiv*, abs/2311.13443, 2023.

A Full Algorithm and Implementation Details

This appendix provides the implementation details omitted from Section 4. We separate the learned components from the training-time target construction procedure. The implementation contains an online double critic, a separately parameterized behavior model, and the final drifting actor (Deng et al., 2026). The behavior model and critic are used to construct detached value-guided target actions, while the final actor is optimized only through the implemented drifting loss.

Unless stated otherwise, all quantities used by the target constructor are treated as stop-gradient supervision for the final actor.

A.1 CRITIC OBJECTIVE

We use an online double critic Q_{ϕ_1}, Q_{ϕ_2} together with target critics $Q_{\bar{\phi}_1}, Q_{\bar{\phi}_2}$, following common double-critic actor–critic practice in offline RL (Fujimoto & Gu, 2021; Park et al., 2025b). For each next state s', the critic target is constructed from actions sampled by the separately learned behavior model rather than by the final policy.

Specifically, we draw K_Q behavior actions

$$\tilde{a}'_k = \mathrm{clip}\big(\beta(s', z_k), -1, 1\big), \qquad z_k \sim \mathcal{N}(0, I), \qquad k = 1, \dots, K_Q. \tag{17}$$

Their target-critic scores are aggregated as

$$\bar{Q}(s', a) = \min_{e \in \{1,2\}} Q_{\bar{\phi}_e}(s', a), \tag{18}$$

and the highest-scoring behavior proposal is selected,

$$k^\star = \arg\max_k \bar{Q}(s', \tilde{a}'_k), \qquad a' = \mathrm{sg}\big(\tilde{a}'_{k^\star}\big). \tag{19}$$

The Bellman target is

$$y = r + \gamma m \, \bar{Q}(s', a'), \tag{20}$$

where m denotes the bootstrap mask. The critic objective is

$$\mathcal{L}_Q = \frac{1}{2B} \sum_{b=1}^{B} \sum_{e=1}^{2} \left(Q_{\phi_e}(s_b, a_b) - y_b\right)^2. \tag{21}$$

The target critics are updated by Polyak averaging,

$$\bar{\phi}_e \leftarrow \tau_Q \phi_e + (1 - \tau_Q)\bar{\phi}_e. \tag{22}$$

A.2 STANDARD DRIFTING OBJECTIVE

We first restate the standard drifting objective used throughout our implementation (Deng et al., 2026). Let $X = \{x_i\}_{i=1}^G$ denote the current generator outputs used for prediction, and $\mathcal{P} = \{p_j\}_{j=1}^{J_+}$ denote positive target samples with weights $\omega = \{\omega_j\}_{j=1}^{J_+}$. Let $\mathcal{Y}^- = \{y_k^-\}_{k=1}^{J_-}$ denote the negative references used for repulsion, where $J_+ = |\mathcal{P}|$ and $J_- = |\mathcal{Y}^-|$. All positive and negative references and positive weights are treated as stop-gradient inputs. We also define

$$\bar{x}_i = \mathrm{sg}(x_i), \tag{23}$$

and form the reference set

$$\mathcal{Y} = \mathcal{Y}^- \cup \mathcal{P}, \qquad w(y) = \begin{cases} 1, & y = y_k^-, \\ \omega_j, & y = p_j. \end{cases} \tag{24}$$

Following the scale-normalized implementation of drifting, we compute the detached distance scale

$$s = \mathrm{sg}\left[\max\left(\frac{\sum_{i=1}^G \sum_{y \in \mathcal{Y}} w(y) \|\bar{x}_i - y\|_2}{G \sum_{y \in \mathcal{Y}} w(y)}, \epsilon_s\right)\right], \qquad c = \frac{s}{\sqrt{d_a}}, \tag{25}$$

where d_a is the action dimension. For each reference $y \in \mathcal{Y}$, define the normalized pairwise distance

$$D_{iy} = \frac{\|\bar{x}_i - y\|_2}{s}. \tag{26}$$

For each radius $R \in \mathcal{R}$, the affinity logits are

$$L_{iy}^{(R)} = -\frac{D_{iy}}{R}, \tag{27}$$

where the self-correspondence is masked whenever $\bar{x}_i$ appears as its own negative reference. Drifting combines row- and column-normalized affinities as

$$A_{iy}^{(R)} = w(y) \sqrt{\max\left(\mathrm{softmax}_y\left(L^{(R)}\right)_{iy} \mathrm{softmax}_i\left(L^{(R)}\right)_{iy}, \epsilon_A\right)}. \tag{28}$$

We write $A_{ik}^{-,(R)}$ for affinities to the negative references $y_k^- \in \mathcal{Y}^-$ and $A_{ij}^{+,(R)}$ for affinities to the positive targets p_j. Their signed coefficients are

$$C_{ik}^{-,(R)} = -A_{ik}^{-,(R)} \sum_{j=1}^{J_+} A_{ij}^{+,(R)}, \qquad C_{ij}^{+,(R)} = A_{ij}^{+,(R)} \sum_{k=1}^{J_-} A_{ik}^{-,(R)}. \tag{29}$$

The resulting radius-specific drifting field is

$$F_i^{(R)} = \sum_{k=1}^{J_-} C_{ik}^{-,(R)} \frac{y_k^- - \bar{x}_i}{c} + \sum_{j=1}^{J_+} C_{ij}^{+,(R)} \frac{p_j - \bar{x}_i}{c}. \tag{30}$$

Each field is normalized by its root-mean-square magnitude,

$$\widehat{F}_i^{(R)} = \frac{F_i^{(R)}}{\sqrt{\max\left(\mathrm{mean}_{i,d}\left[\left(F^{(R)}\right)^2\right], \epsilon_F\right)}}. \tag{31}$$

Finally, the standard drifting objective is

$$\mathcal{L}_{\mathrm{drift}}\left(X; \mathcal{P}, \mathcal{Y}^-, \omega\right) = \mathbb{E}_{i\sim\mathrm{Unif}(\{1,\dots,G\})}\left[\frac{1}{d_a}\left\|\frac{x_i}{c} - \mathrm{sg}\left(\frac{\bar{x}_i}{c} + \sum_{R\in\mathcal{R}} \widehat{F}_i^{(R)}\right)\right\|_2^2\right]. \tag{32}$$

A.3 Behavior Model Objective

QGD uses a separate behavior model β to generate behavior-conditioned anchors for target construction. The behavior model is independent of the deployed policy π_θ and is trained only on the offline dataset.

Given a state s, the behavior model generates

$$x_i^\beta = \beta(s, z_i), \qquad z_i \sim \mathcal{N}(0, I), \qquad i = 1, \dots, G_\beta, \tag{33}$$

and we denote the generated set by $X^\beta = \{x_i^\beta\}_{i=1}^{G_\beta}$. The behavior model uses its own stopped samples as negative references,

$$\mathcal{Y}_\beta^- = \{\mathrm{sg}(x_i^\beta)\}_{i=1}^{G_\beta}. \tag{34}$$

The dataset action $a_\mathcal{D}$ serves as the positive target, so the behavior model is trained with

$$\mathcal{L}_\beta = \mathcal{L}_{\mathrm{drift}}\left(X^\beta; \{a_\mathcal{D}\}, \mathcal{Y}_\beta^-, \mathbf{1}\right). \tag{35}$$

Here, $\mathbf{1}$ denotes the unit weight assigned to the dataset target. The full form of $\mathcal{L}_{\mathrm{drift}}$ is given in Appendix A.2.

A.4 Per-Anchor Local Refinement and Target Construction

For each state s, QGD first samples M behavior-conditioned anchors,

$$a_m = \mathrm{clip}(\beta(s, \xi_m), -1, 1), \qquad \xi_m \sim \mathcal{N}(0, I), \qquad m = 1, \dots, M. \tag{36}$$

Each anchor maintains an independent local population of N particles. The initial population is

$$a_{m,n}^0 = \mathrm{clip}\big(a_m + \sigma_0 \epsilon_{m,n}^0, -1, 1\big), \qquad \epsilon_{m,n}^0 \sim \mathcal{N}(0, I), \tag{37}$$

with the first particle replaced by the anchor, $a_{m,1}^0 = a_m$.

For each refinement round $t = 0, \dots, T_r - 1$, we first evaluate detached critic scores and normalize them independently within each anchor:

$$q_{m,n}^t = \mathrm{sg}\left[\min_{e\in\{1,2\}} Q_{\phi_e}(s, a_{m,n}^t)\right], \qquad \hat{q}_{m,n}^t = \frac{q_{m,n}^t - \mathrm{mean}_u\, q_{m,u}^t}{\mathrm{std}_u\, q_{m,u}^t + \epsilon_Q}. \tag{38}$$

The corresponding normalized refinement weights are

$$\omega_{m,n}^{t} = \frac{\exp\left(\hat{q}_{m,n}^{t}/\tau_{\text{refine}}\right)}{\sum_{u=1}^{N}\exp\left(\hat{q}_{m,u}^{t}/\tau_{\text{refine}}\right)}. \tag{39}$$

Each anchor then forms its own weighted local mean, diagonal variance, and clipped standard deviation:

$$\begin{aligned}
\mu_m^{t+1} &= \sum_{n=1}^{N} \omega_{m,n}^{t} a_{m,n}^{t}, \\
v_m^{t+1} &= \sum_{n=1}^{N} \omega_{m,n}^{t} \left(a_{m,n}^{t} - \mu_m^{t+1}\right)^2, \\
\widetilde{\sigma}_m^{t+1} &= \operatorname{clip}\left(\sqrt{\max(v_m^{t+1}, \epsilon_v)}, \sigma_{\min}, \sigma_{\max}\right).
\end{aligned} \tag{40}$$

Here, the square, maximum, square root, and clipping operations are applied elementwise.

The next local population is sampled independently for each anchor:

$$\begin{aligned}
a_{m,n}^{t+1} &= \operatorname{clip}\left(\mu_m^{t+1} + \widetilde{\sigma}_m^{t+1} \odot \epsilon_{m,n}^{t+1}, -1, 1\right), \qquad \epsilon_{m,n}^{t+1} \sim \mathcal{N}(0, I), \\
a_{m,1}^{t+1} &= \mu_m^{t+1}.
\end{aligned} \tag{41}$$

After the final refinement round, each anchor contributes its refined weighted mean. Defining $\mu_m \equiv \mu_m^{T_r}$, the resulting candidate set is

$$\mathcal{C}(s) = \{\mu_m\}_{m=1}^{M}. \tag{42}$$

Here, μ_m corresponds to the refined candidate $c_{m,s}$ and $\mathcal{C}(s)$ corresponds to $\mathcal{C}_s$ in the main text.

The refined candidates are ranked using detached critic scores,

$$q_m = \operatorname{sg}\left[\min_{e\in\{1,2\}} Q_{\phi_e}(s, \mu_m)\right], \qquad m^\star = \arg\max_m q_m. \tag{43}$$

Following the implementation, the highest-scoring candidate $\mu_{m^\star}$ is removed before positive selection. The highest-scoring candidates among the remainder are retained as

$$\mathcal{P}_s^{\text{ref}} = \operatorname{TopK}_{K_+}\left(\mathcal{C}(s) \setminus \{\mu_{m^\star}\}; \{q_m\}_{m=1}^{M}\right). \tag{44}$$

This Top-K operation is used only for the final selection of refined candidates; the per-anchor refinement itself uses all particles.

The dataset action is then added to form the final positive target set,

$$\mathcal{P}_s = \{a_{\mathcal{D}}\} \cup \mathcal{P}_s^{\text{ref}}. \tag{45}$$

Each refined target action $p \in \mathcal{P}_s^{\text{ref}}$ is assigned a detached critic-derived score. We normalize the critic scale within the refined target set as

$$q(p) = \operatorname{sg}\left[\min_{e\in\{1,2\}} Q_{\phi_e}(s, p)\right], \qquad \lambda_s = \left(\frac{1}{|\mathcal{P}_s^{\text{ref}}|}\sum_{\tilde{p}\in\mathcal{P}_s^{\text{ref}}} |q(\tilde{p})| + \epsilon_w\right)^{-1}. \tag{46}$$

The normalized target weights and the mean-one weights used by drifting are

$$\rho(p) = \frac{\exp(\lambda_s q(p)/\tau_P)}{\sum_{\tilde{p}\in\mathcal{P}_s^{\text{ref}}}\exp(\lambda_s q(\tilde{p})/\tau_P)}, \qquad \omega_Q(p) = |\mathcal{P}_s^{\text{ref}}|\,\rho(p). \tag{47}$$

The dataset action is assigned the domain-dependent weight $w_{\mathcal{D}}$ reported in Table 4. We define the complete positive weight function and weighted target set as

$$\omega(p) = \begin{cases} w_{\mathcal{D}}, & p = a_{\mathcal{D}}, \\ \omega_Q(p), & p \in \mathcal{P}_s^{\text{ref}}, \end{cases} \qquad \mathcal{T}_s = \{(p, \omega(p)) : p \in \mathcal{P}_s\}. \tag{48}$$

Finally, the additional negative references are selected from the initial behavior anchors using detached critic scores,

$$\mathcal{N}_s = \operatorname{BottomK}_{K_-}\left(\{a_m\}_{m=1}^{M}; \overline{Q}_\phi(s, \cdot)\right). \tag{49}$$

A.5 Policy Objective

Given a state s, the final policy generates

$$x_i^\pi = \pi_\theta(s, z_i), \qquad z_i \sim \mathcal{N}(0, I), \qquad i = 1, \dots, G_\pi, \tag{50}$$

with generated sample set $X^\pi = \{x_i^\pi\}_{i=1}^{G_\pi}$. As in the main text, let $\mathcal{N}_s$ contain the low-value initial behavior samples selected by detached critic scores. The policy uses the negative references

$$\mathcal{Y}_s^- = \{\mathrm{sg}(x_i^\pi)\}_{i=1}^{G_\pi} \cup \mathrm{sg}(\mathcal{N}_s). \tag{51}$$

Given the target-action set $\mathcal{P}_s$ and its weights ω constructed in Appendix A.4, the policy is trained with the standard drifting objective

$$\mathcal{L}_\pi = \mathcal{L}_{\mathrm{drift}}\left(X^\pi; \mathcal{P}_s, \mathcal{Y}_s^-, \omega\right). \tag{52}$$

B Theoretical Details

Let $\mathcal{Q}$ denote a Gaussian family with fixed covariance, $\mathcal{Q} = \{q_\theta(a|s) = \mathcal{N}(a; \mu_\theta(s), \Sigma)\}$. Given an initial distribution $q^0 \in \mathcal{Q}$, define the refinement sequence $\{q^t\}_{t=0}^T$ by

$$q^{t+1} = \arg\max_{q \in \mathcal{Q}} \left\{\mathbb{E}_q[Q_\phi(s,a)] - \alpha D_{\mathrm{KL}}(q \| q^t)\right\}. \tag{53}$$

Assume that each update satisfies

$$D_{\mathrm{KL}}(q^{t+1} \| q^t) \le \epsilon_t. \tag{54}$$

Then the corresponding unconstrained optimum is the Boltzmann distribution

$$\tilde{q}^{t+1}(a|s) = \frac{q^t(a|s)\exp(Q_\phi(s,a)/\alpha)}{Z_t(s)}, \tag{55}$$

and the Gaussian refinement q^{t+1} is its KL projection onto $\mathcal{Q}$. Furthermore, the finite-step refinement satisfies

$$D_{\mathrm{KL}}(q^T \| q^0) \le C \sum_{t=0}^{T-1} \epsilon_t, \tag{56}$$

where C is determined by the regularity of the refinement path. Therefore, q^T is a Gaussian approximation of the Q-guided Boltzmann distribution while remaining bounded from the initial proposal.

Proof. For each refinement step, consider the unconstrained variational objective

$$\mathcal{J}(q) = \mathbb{E}_q[Q_\phi(s,a)] - \alpha D_{\mathrm{KL}}(q \| q^t).$$

Expanding the KL divergence gives

$$\mathcal{J}(q) = \int q(a|s)\left[Q_\phi(s,a) - \alpha \log q(a|s) + \alpha \log q^t(a|s)\right] da. \tag{57}$$

Introducing the normalization constraint $\int q(a|s)da = 1$, the corresponding Lagrangian is $\mathcal{L}(q) = \mathcal{J}(q) + \eta\left(\int q(a|s)da - 1\right)$. Taking the functional derivative with respect to $q(a|s)$ yields

$$0 = Q_\phi(s,a) - \alpha(\log q(a|s) + 1) + \alpha \log q^t(a|s) + \eta. \tag{58}$$

Rearranging gives

$$\log q(a|s) = \log q^t(a|s) + \frac{Q_\phi(s,a)}{\alpha} + C, \tag{59}$$

where C is the normalization constant. Therefore,

$$\tilde{q}^{t+1}(a|s) = \frac{q^t(a|s)\exp(Q_\phi(s,a)/\alpha)}{Z_t(s)}.$$

Since the practical refinement is restricted to the Gaussian family $\mathcal{Q}$, the resulting distribution is obtained by the KL projection

$$q^{t+1} = \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}(\tilde{q}^{t+1} \| q).$$

Finally, under the bounded refinement condition

$$D_{\mathrm{KL}}(q^{t+1} \| q^t) \le \epsilon_t$$

and the local regularity of the refinement path, the accumulated deviation from the initial proposal satisfies

$$D_{\mathrm{KL}}(q^T \| q^0) \le C \sum_{t=0}^{T-1} \epsilon_t.$$

Thus the finite-step Gaussian refinement remains a bounded KL deviation from the initial proposal while approximating the Q-guided Boltzmann distribution.

□

B.1 Proof of Target Property

Proof. For each behavior anchor a_m, by the finite-step refinement assumption in Proposition 1, the local refinement distribution q_m remains close to the corresponding local Boltzmann target

$$q_m^*(a|s) = \frac{1}{Z_m(s)} q_m^0(a|s) \exp\left(\frac{Q_\phi(s,a)}{\alpha}\right). \tag{60}$$

Therefore, by the convexity of total variation distance, the mixture of finite-step refinements satisfies

$$\begin{aligned} & D_{\mathrm{TV}}\left(\frac{1}{M}\sum_{m=1}^{M} q_m(\cdot|s), \frac{1}{M}\sum_{m=1}^{M} q_m^*(\cdot|s)\right) \\ & \le \frac{1}{M}\sum_{m=1}^{M} D_{\mathrm{TV}}(q_m(\cdot|s), q_m^*(\cdot|s)) \\ & \le \epsilon_{\mathrm{ref}}, \end{aligned} \tag{61}$$

where ϵ_{ref} denotes the finite-step refinement error assumed in Proposition 1.

Next, consider the mixture of ideal local Boltzmann refinements. Define the empirical mixture of local proposals as

$$\bar{q}^0(a|s) = \frac{1}{M}\sum_{m=1}^{M} q_m^0(a|s). \tag{62}$$

Since the local proposals are initialized from behavior samples, their mixture approximates the behavior distribution:

$$D_{\mathrm{TV}}(\bar{q}^0(\cdot|s), \pi_\beta(\cdot|s)) \le \epsilon_\beta. \tag{63}$$

The ideal local refinement mixture is

$$\bar{q}^*(a|s) = \frac{1}{M}\sum_{m=1}^{M} \frac{q_m^0(a|s)\exp(Q_\phi(s,a)/\alpha)}{Z_m(s)}. \tag{64}$$

Let

$$Z(s) = \int \pi_\beta(a|s)\exp(Q_\phi(s,a)/\alpha)da \tag{65}$$

denote the global Boltzmann normalization factor. Since each local proposal is centered around a behavior sample and the local partition functions satisfy

$$|Z_m(s) - Z(s)| \le \epsilon_Z, \tag{66}$$

with $Z_m(s), Z(s) \geq \delta$, we have

$$\left| \frac{1}{Z_m(s)} - \frac{1}{Z(s)} \right| = \frac{|Z_m(s) - Z(s)|}{Z_m(s)Z(s)} \leq \frac{\epsilon_Z}{\delta^2}. \tag{67}$$

Hence replacing each local normalization factor by the global normalization introduces at most

$$D_{\mathrm{TV}}\left(\bar{q}^*(\cdot|s), \frac{\bar{q}^0(\cdot|s)\exp(Q_\phi(s,\cdot)/\alpha)}{Z(s)} \right) \leq C\frac{\epsilon_Z}{\delta^2} \tag{68}$$

error, where

$$C = \frac{1}{2}\sup_a \exp(Q_\phi(s,a)/\alpha). \tag{69}$$

Furthermore, the Boltzmann transformation of the behavior mixture gives

$$D_{\mathrm{TV}}\left(\frac{\bar{q}^0(\cdot|s)\exp(Q_\phi(s,\cdot)/\alpha)}{Z(s)}, \frac{\pi_\beta(\cdot|s)\exp(Q_\phi(s,\cdot)/\alpha)}{Z(s)} \right) \leq \epsilon_\beta. \tag{70}$$

Combining the above bounds using the triangle inequality yields

$$\begin{aligned} &D_{\mathrm{TV}}(p_{\mathrm{QGD}}(\cdot|s), \pi^*(\cdot|s)) \\ &\leq D_{\mathrm{TV}}\left(p_{\mathrm{QGD}}, \frac{1}{M}\sum_m q_m^* \right) \\ &\quad + D_{\mathrm{TV}}\left(\frac{1}{M}\sum_m q_m^*, \pi^* \right) \\ &\leq \epsilon_{\mathrm{ref}} + \epsilon_\beta + C\frac{\epsilon_Z}{\delta^2}. \end{aligned} \tag{71}$$

Therefore, the target distribution constructed by QGD remains close to the behavior-regularized Boltzmann improvement distribution while being represented as a mixture of multiple local refinements. This mixture structure preserves multiple high-value action regions. □

C Experiment Details

C.1 OGBench

OGBench (Park et al., 2025a) is a benchmark for offline reinforcement learning that provides diverse locomotion and robotic manipulation environments with fixed offline datasets. We use the reward-based single-task variants, denoted by the `-singletask` suffix, in which each environment contains five fixed evaluation tasks corresponding to five different target goals. Our evaluation includes five state-based environments—Cube-Double, Cube-Triple, Scene, Puzzle-3 $\times$ 3, and Puzzle-4 $\times$ 4—resulting in a total of 25 tasks. These environments cover object manipulation, sequential reasoning, and combinatorial generalization.

Cube. The Cube environments evaluate basic robotic pick-and-place manipulation. A robot arm interacts with colored cubes and must arrange them into specified target configurations. We use Cube-Double and Cube-Triple, which contain two and three movable cubes, respectively. The five evaluation tasks involve different object arrangements and require behaviors such as moving, stacking, or rearranging cubes. We use the `play` datasets, which contain diverse object interactions rather than demonstrations tailored to a particular evaluation task. Consequently, the agent must recover reusable manipulation skills from the offline data and compose them to solve the fixed downstream task. The corresponding datasets are `cube-double-play-singletask-task{1,...,5}-v0` and `cube-triple-play-singletask-task{1,...,5}-v0`.

Scene. Scene is a long-horizon manipulation environment containing several interacting objects, including a cube, a drawer, a window, and buttons that control the locking states of other objects. The five evaluation tasks specify different desired scene configurations and typically require multiple manipulation skills to be executed in a particular sequence. For example, completing a task may require unlocking an object before manipulating it and subsequently restoring another part of the scene. The `play` dataset is collected from diverse interactions with the objects rather than task-specific demonstrations, making successful policy learning dependent on sequential composition of behaviors over relatively long horizons. We evaluate `scene-play-singletask-task{1,...,5}-v0`.

Puzzle. The Puzzle environments instantiate the Lights Out puzzle as a continuous-control robotic manipulation problem. The robot presses buttons arranged on a grid, where pressing one button changes both that button and its neighboring buttons. The objective is to reach a specified target configuration. We consider the 3×3 and 4×4 variants, each with five fixed evaluation tasks of varying difficulty. The datasets consist of diverse sequences of button presses and do not directly provide solutions to every target configuration. As the grid size increases, the number of possible configurations grows combinatorially, making these environments particularly challenging for long-horizon value estimation and compositional generalization. We evaluate `puzzle-3x3-play-singletask-task{1,...,5}-v0` and `puzzle-4x4-play-singletask-task{1,...,5}-v0`.

Rewards and evaluation. Following the standard reward-based OGBench protocol, each of the five single-task variants within an environment corresponds to a fixed evaluation goal. The manipulation environments provide task progress through their benchmark-defined reward functions. We evaluate each task using the environment success criterion and aggregate results across the five single-task variants when reporting environment-level performance.

C.2 D4RL

D4RL (Fu et al., 2020) is a widely used benchmark for offline reinforcement learning that provides fixed datasets with varying levels of coverage and behavior quality. We evaluate QGD on 12 D4RL Adroit dexterous-manipulation tasks. These environments test offline policy learning under sparse rewards, long-horizon credit assignment, heterogeneous behavior data, and high-dimensional continuous control.

Adroit. The Adroit suite evaluates dexterous manipulation with a high-dimensional anthropomorphic hand. We consider four tasks: Pen, Door, Hammer, and Relocate, covering object reorientation, articulated-object manipulation, tool use, and object relocation, respectively. For each task, we evaluate the `human`, `cloned`, and `expert` dataset regimes, resulting in twelve tasks: `pen-human-v1`, `pen-cloned-v1`, `pen-expert-v1`, `door-human-v1`, `door-cloned-v1`, `door-expert-v1`, `hammer-human-v1`, `hammer-cloned-v1`, `hammer-expert-v1`, `relocate-human-v1`, `relocate-cloned-v1`, and `relocate-expert-v1`. These datasets span different behavior distributions and data-quality regimes, providing a challenging test of offline learning in high-dimensional continuous-control problems.

Evaluation. Following the standard D4RL evaluation protocol, we report normalized returns for all D4RL tasks. Higher normalized return indicates better policy performance relative to the benchmark reference scores.

D Reproducibility Details

Our implementation is based on JAX/Flax. All experiments were conducted on a single NVIDIA RTX 4090 GPU.

D.1 Training Procedure

Training uses minibatches sampled uniformly from the fixed offline dataset. Algorithm 1 summarizes the implementation of the complete training procedure. The critic, behavior-model, and policy

losses are evaluated before a single joint optimizer update. Target construction uses the current behavior-model and critic parameters, and all critic-derived targets and weights are detached from the policy optimization graph.

Algorithm 1 Training procedure of QGD.

Require: Offline dataset $\mathcal{D}$
Initialize critics, target critics, behavior model β, and policy π_θ
while training **do**
  Sample a minibatch from $\mathcal{D}$
  Compute critic loss $\mathcal{L}_Q$ as in Eq. 21
  Compute behavior-model loss $\mathcal{L}_\beta$ as in Eq. 35
  Sample behavior-conditioned anchors from β
  Perform per-anchor zeroth-order refinement using detached scalar critic evaluations
  Construct the weighted target set $\mathcal{T}_s$ as in Eq. 48
  Compute policy loss $\mathcal{L}_\pi$ as in Eq. 52
  Compute $\mathcal{L}_{\text{total}} = \mathcal{L}_Q + \mathcal{L}_\beta + \mathcal{L}_\pi$
  Apply one joint optimizer update
  Update target critics by Polyak averaging
end while

D.2 Architectures and Hyperparameters

We parameterize the critic, behavior model, and final policy with separate networks. The critic is implemented as a two-head ensemble over observation–action inputs, while the behavior model and policy use the same noise-conditioned actor architecture with independent parameters. For visual tasks, each component uses its own encoder. The architectural specifications are summarized in Table 2, and all optimization, target-construction, and drifting hyperparameters are listed in Table 3. Table 4 lists domain- and task-dependent baseline hyperparameters and discount factors alongside the dataset-action weight $w_{\mathcal{D}}$ for QGD.

Table 2: Network architectures used by QGD.

Architecture	**Specification**
Critic	Double Q ensemble
Hidden dimensions	$[512, 512, 512, 512]$
Activation	GELU
Layer normalization	Yes
Behavior model	ActorVectorField
Hidden dimensions	$[512, 512, 512, 512]$
Activation	GELU
Layer normalization	No
Policy	ActorVectorField
Hidden dimensions	$[512, 512, 512, 512]$
Activation	GELU
Layer normalization	No
Visual encoder	IMPALA-small
Encoder sharing	Independent across critic, behavior model, and policy

Table 3: Implementation hyperparameters for QGD.

Hyperparameter	**Value**
Learning rate	3×10^{-4}
Optimizer	Adam
Batch size	256
Gradient steps	10^6
Target update rate τ_Q	0.005
Critic aggregation	Minimum
Critic target proposals K_Q	16
Behavior anchors M	8
Particles per anchor N	8
Refinement rounds T_r	4
Initial scale σ_0	0.10
Minimum scale $\sigma_{\min}$	0.05
Maximum scale $\sigma_{\max}$	0.30
Refined targets retained	2
Target temperature τ_P	10
Drifting temperature τ_{drift}	$\{0.05, 0.2, 1.0\}$
Self-mask constant	100

Table 4: **Domain- and task-dependent hyperparameters.** Baseline settings are transcribed from the reference papers. "-" indicates the results were taken from prior work (Park et al., 2025b). The discount factor γ is shared for all methods in each domain.

Domain or task	Discount γ	IQL α	ReBRAC (α_1, α_2)	FBRAC α	IFQL N / κ	QAM τ	VGF $(L_{\text{train}}, \epsilon)$	FQL α	Shortcut FQL: α	MeanFlow FQL: α	QGD $w_{\mathcal{D}}$
cube-double-play	0.995	0.3	$(0.1, 0)$	100	32	1	$(1, 0.05)$	300	300	300	1.5
cube-triple-play	0.995	10	$(0.03, 0)$	100	32	3	–	300	300	300	1.5
scene-play	0.99	10	$(0.1, 0.001)$	100	32	–	$(1, 0.1)$	300	300	300	1
puzzle-3x3-play	0.99	10	$(0.3, 0.001)$	100	32	–	$(5, 0.1)$	1000	1000	1000	1
puzzle-4x4-play	0.99	3	$(0.3, 0.01)$	300	32	–	$(5, 0.1)$	1000	1000	1000	1.5
visual-antmaze-medium	0.99	1	$(0.01, 0.003)$	100	32	–	–	100	100	100	1
antmaze-large	0.99	–	$(0.01, 0)$	0.1	0.9	10	$(5, 0.1)$	3	3	3	1
humanoidmaze-medium	0.995	–	$(0.01, 0)$	30	0.7	3	$(1, 0.05)$	30	30	30	1
pen-human	0.99	–	$(0.1, 0.5)$	30000	32	–	–	10000	10000	10000	1.5
pen-cloned	0.99	–	$(0.05, 0.5)$	10000	32	–	–	10000	10000	10000	1.5
pen-expert	0.99	–	$(0.01, 0.01)$	30000	32	–	–	3000	3000	3000	1.5
door-human	0.99	–	$(0.1, 0.1)$	30000	32	–	–	30000	30000	30000	1.5
door-cloned	0.99	–	$(0.01, 0.1)$	10000	32	–	–	30000	30000	30000	1.5
door-expert	0.99	–	$(0.05, 0.01)$	30000	32	–	–	30000	30000	30000	1.5
hammer-human	0.99	–	$(0.01, 0.5)$	30000	128	–	–	30000	30000	30000	1.5
hammer-cloned	0.99	–	$(0.1, 0.5)$	10000	32	–	–	10000	10000	10000	1.5
hammer-expert	0.99	–	$(0.01, 0.01)$	30000	32	–	–	30000	30000	30000	1.5
relocate-human	0.99	–	$(0.1, 0.01)$	30000	32	–	–	10000	10000	10000	1.5
relocate-cloned	0.99	–	$(0.1, 0.01)$	3000	32	–	–	30000	30000	30000	1.5
relocate-expert	0.99	–	$(0.05, 0.01)$	30000	32	–	–	30000	30000	30000	1.5

D.3 Evaluation Protocol

We follow the benchmark-specific evaluation metrics used in the main experiments. OGBench tasks are evaluated using task success statistics returned by the environment, while D4RL tasks are evaluated using normalized return. Each evaluation uses 50 episodes.

The policy remains stochastic during evaluation. At each environment step, a fresh Gaussian latent is sampled and the action is generated by a single forward pass of the policy:

$$z_t \sim \mathcal{N}(0, I), \qquad a_t = \text{clip}\left(\pi_\theta(s_t, z_t), -1, 1\right). \tag{72}$$

The behavior model, critic, and QGD target-construction procedure are not used during evaluation. Episode horizons follow the termination and truncation rules of the corresponding environments.

For D4RL tasks, normalized return is computed using the benchmark-provided normalization:

$$R_{\text{norm}} = 100 \times \text{NormalizedScore}\left(R_{\text{episode}}\right). \tag{73}$$

D.4 Baseline Implementations

Conventional Policies. We include behavioral cloning (BC), Implicit Q-Learning (IQL) (Kostrikov et al., 2022), and ReBRAC (Tarasov et al., 2023).

Behavior Cloning (BC). BC directly fits the behavior distribution represented by the offline dataset without explicitly optimizing reward or cost. For a stochastic policy π_θ, the standard maximum-likelihood objective is

$$\mathcal{L}_{\text{BC}}(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log \pi_\theta(a \mid s)\right]. \tag{74}$$

BC therefore serves as a behavioral reference that does not explicitly perform reward maximization or constraint-aware policy improvement.

Implicit Q-Learning (IQL). IQL (Kostrikov et al., 2022) performs offline policy improvement without explicitly querying out-of-distribution actions. It learns a state-value function by expectile regression over dataset actions,

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_\tau\left(Q_\phi(s,a) - V_\psi(s)\right)\right], \tag{75}$$

where

$$L_\tau(u) = |\tau - \mathbf{1}\{u < 0\}|\, u^2. \tag{76}$$

The critic is trained with the temporal-difference objective

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left(r + \gamma V_\psi(s')\right)\right)^2\right], \tag{77}$$

and the policy is extracted through advantage-weighted behavioral regression,

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\beta\left(Q_\phi(s,a) - V_\psi(s)\right)\right)\log \pi_\theta(a \mid s)\right]. \tag{78}$$

IQL therefore serves as an in-sample offline RL baseline that converts learned value information into policy improvement through advantage-weighted regression.

ReBRAC. ReBRAC (Tarasov et al., 2023) is a behavior-regularized actor–critic method that combines value maximization with explicit regularization toward dataset actions. Its actor objective is

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[Q_\phi(s,\pi_\theta(s)) - \beta_1 \left\|\pi_\theta(s) - a\right\|_2^2\right], \tag{79}$$

where β_1 controls the strength of actor behavior regularization. ReBRAC additionally regularizes the temporal-difference target using the next dataset action $\hat{a}'$:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s',\hat{a}')\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left[r + \gamma\left(Q_{\bar{\phi}}(s',a') - \beta_2\left\|a' - \hat{a}'\right\|_2^2\right)\right]\right)^2\right], \tag{80}$$

where $a' = \pi_\theta(s')$, $Q_{\bar{\phi}}$ denotes the target critic, and β_2 independently controls critic-side behavior regularization. ReBRAC therefore serves as a conventional behavior-regularized offline RL baseline in which learned value information directly participates in actor optimization while policy deviation from the offline behavior is explicitly penalized.

Multi-Step and Search-Based Generative Policies. We include FBRAC and IFQL (Park et al., 2025b), Q-Learning with Adjoint Matching (QAM) (Li & Levine, 2026), and Value Gradient Flow (VGF) (Xu et al., 2026).

Flow Behavior-Regularized Actor-Critic (FBRAC). FBRAC is the flow counterpart of Diffusion-QL (Wang et al., 2023) considered in FQL (Park et al., 2025b). Let $\pi_v(s, z)$ denote the action obtained by integrating the flow policy from noise $z \sim \mathcal{N}(0, I)$. FBRAC optimizes the flow policy with a direct critic objective together with flow-matching behavior regularization,

$$\mathcal{L}_{\mathrm{FBRAC}}(v) = -\mathbb{E}_{s\sim\mathcal{D},\, z\sim\mathcal{N}(0,I)} \left[Q_\phi(s, \pi_v(s, z))\right] + \alpha\, \mathcal{L}_{\mathrm{FM}}(v), \tag{81}$$

where $\mathcal{L}_{\mathrm{FM}}$ denotes the standard flow-matching objective on offline actions and α controls the strength of behavior regularization. Since $\pi_v(s, z)$ is obtained through iterative flow integration, the Q-maximization term requires backpropagation through time through the flow-generation process. FBRAC therefore represents a multi-step flow-policy baseline based on direct critic optimization.

Implicit Flow Q-Learning (IFQL). IFQL is a flow-based policy-extraction baseline considered by FQL (Park et al., 2025b). It combines in-sample value learning with an expressive flow-based behavior model. At inference, the flow model generates multiple candidate actions from the learned behavior distribution, and scalar critic evaluations are used to select among these candidates. Given candidates

$$a^{(i)} \sim \pi_{\mathrm{flow}}(\cdot \mid s), \qquad i = 1, \ldots, N, \tag{82}$$

the selected action is

$$a^* = \arg\max_{a^{(i)}} Q_\phi(s, a^{(i)}). \tag{83}$$

IFQL therefore serves as a search-based generative baseline in which value information is used for candidate selection rather than direct optimization of the flow policy.

Q-Learning with Adjoint Matching (QAM). QAM (Li & Levine, 2026) optimizes an expressive flow policy using first-order critic information through adjoint matching. Let f_θ denote the policy flow and f_β the behavior flow. For a sampled flow trajectory $\{a_t\}_{t\in[0,1]}$, QAM initializes the terminal adjoint state with the critic action gradient,

$$\tilde{g}_1 = -\tau \nabla_{a_1} Q_\phi(s, a_1), \tag{84}$$

and propagates the lean adjoint state backward according to

$$\mathrm{d}\tilde{g}_t = -\nabla_{a_t} \left[2 f_\beta(s, a_t, t) - \frac{a_t}{t}\right] \tilde{g}_t \,\mathrm{d}t. \tag{85}$$

The policy is then optimized with the adjoint-matching objective

$$\mathcal{L}_{\mathrm{QAM}}(\theta) = \mathbb{E}_{s\sim\mathcal{D},\, \{a_t\}} \left[\int_0^1 \left\| \frac{2\left(f_\theta(s, a_t, t) - f_\beta(s, a_t, t)\right)}{\sigma_t} + \sigma_t \tilde{g}_t \right\|_2^2 \mathrm{d}t\right]. \tag{86}$$

QAM therefore provides a first-order critic interface for expressive generative policy optimization without directly backpropagating through the complete multi-step generation process.

Value Gradient Flow (VGF). VGF (Xu et al., 2026) performs behavior-regularized policy improvement through iterative particle transport. Given N initial actions sampled from a learned behavior policy,

$$a_i^{(0)} \sim \hat{\mu}(\cdot \mid s), \qquad i = 1, \ldots, N, \tag{87}$$

the particles are iteratively updated as

$$a_i^{(\ell+1)} = a_i^{(\ell)} + \epsilon\, \phi\left(a_i^{(\ell)}\right), \tag{88}$$

where, in the value-gradient update used for offline RL,

$$\phi(x) = \frac{1}{N} \sum_{j=1}^{N} k\left(a_j^{(\ell)}, x\right) \nabla_{a_j^{(\ell)}} Q_\phi\left(s, a_j^{(\ell)}\right). \tag{89}$$

After L_{test} transport steps, the executed action is selected from the refined particles according to

$$a^* = \arg\max_{a_i^{(L_{\mathrm{test}})}} Q_\phi\left(s, a_i^{(L_{\mathrm{test}})}\right). \tag{90}$$

VGF therefore represents a test-time iterative search procedure in which critic action gradients directly determine particle refinement.

Distilled One-Step Policy. FQL (Park et al., 2025b) learns an expressive flow-based behavior model together with a separate one-step policy for offline policy optimization. Let $\mu_\theta(s, z)$ denote the action produced by the behavior flow and $\mu_\omega(s, z)$ the one-step policy, with $z \sim \mathcal{N}(0, I)$. The one-step policy is trained with the objective

$$\mathcal{L}_{\mathrm{FQL}}(\omega) = \mathbb{E}_{s\sim\mathcal{D},\, z\sim\mathcal{N}(0,I)} \left[-Q_\phi(s, \mu_\omega(s, z)) + \alpha \left\| \mu_\omega(s, z) - \mu_\theta(s, z) \right\|_2^2 \right], \tag{91}$$

where the first term directly maximizes the learned critic and the second distills the one-step policy toward the behavior flow. The behavior flow itself is trained only with the standard flow-matching objective. Since critic optimization is applied to the one-step policy rather than through the iterative flow-generation process, FQL avoids recursive backpropagation through the flow while retaining one-step inference at deployment.

Native One-Step Generative Policies. We additionally evaluate two native one-step generative variants, Shortcut-FQL and MeanFlow-FQL. Both variants are constructed within the FQL-style framework and retain the same critic-learning procedure used by FQL, while replacing the policy parameterization with a native one-step generative model. For Shortcut-FQL, we replace the distillation regularization term in FQL with the Shortcut Model training objective

$$\mathcal{L}_{\mathrm{SC}} = \mathbb{E}_{x_0,x_1,t,d} \left[\left\| \mu_\omega(x_t, t, 0) - (x_1 - x_0) \right\|_2^2 + \left\| \mu_\omega(x_t, t, 2d) - s_{\mathrm{target}} \right\|_2^2 \right], \tag{92}$$

where

$$\mu_{\mathrm{target}} = \frac{\mu_\omega(x_t, t, d) + \mu_\omega(x'_{t+d}, t + d, d)}{2}. \tag{93}$$

The generated action is obtained by the one-step Shortcut generation with the full step size. For MeanFlow-FQL, we replace the distillation regularization term with the MeanFlow objective

$$\mathcal{L}_{\mathrm{MF}} = \mathbb{E} \left[\left\| u_\omega(z_t, r, t) - \mathrm{sg}(u_{\mathrm{target}}) \right\|_2^2 \right], \tag{94}$$

where

$$u_{\mathrm{target}} = v_t - (t - r)(v_t \partial_z u_\omega + \partial_t u_\omega). \tag{95}$$

E Full Evaluation Results

E.1 Offline Evaluation Results

Table 5: **Full offline RL results.** OGBench reports success rates (%), and D4RL reports normalized returns. Values are mean ± standard deviation. We bold values within 95% of the best performance for each task.

	Conventional Policies			Multi-Step / Search Policies				Distilled One-Step Policy	Native One-Step Policies		
Task	BC	IQL	ReBRAC	FBRAC	IFQL	QAM	VGF	FQL	Shortcut-FQL	MeanFlow-FQL	**QGD (Ours)**
cube-double-play-singletask-task1-v0	13 ± 4	17 ± 1	2 ± 3	18 ± 6	28 ± 11	25 ± 7	34 ± 6	44 ± 3	0 ± 0	0 ± 0	**60 ± 14**
cube-double-play-singletask-task2-v0 (*)	0 ± 0	0 ± 0	0 ± 0	8 ± 6	5 ± 1	3 ± 1	34 ± 8	**42 ± 6**	0 ± 0	0 ± 0	**42 ± 3**
cube-double-play-singletask-task3-v0	0 ± 0	0 ± 0	0 ± 0	26 ± 11	5 ± 1	0 ± 0	**39 ± 16**	9 ± 4	0 ± 0	0 ± 0	21 ± 1
cube-double-play-singletask-task4-v0	0 ± 0	0 ± 0	0 ± 0	1 ± 1	0 ± 0	**6 ± 3**	2 ± 0	2 ± 0	0 ± 0	0 ± 0	5 ± 1
cube-double-play-singletask-task5-v0	0 ± 0	0 ± 0	0 ± 0	1 ± 1	18 ± 3	14 ± 11	29 ± 7	31 ± 1	0 ± 0	0 ± 0	**46 ± 14**
cube-triple-play-singletask-task1-v0 (*)	0 ± 0	4 ± 1	3 ± 3	6 ± 5	38 ± 20	0 ± 0	0 ± 0	28 ± 19	0 ± 0	0 ± 0	**42 ± 3**
cube-triple-play-singletask-task2-v0	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
cube-triple-play-singletask-task3-v0	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
cube-triple-play-singletask-task4-v0	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
cube-triple-play-singletask-task5-v0	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
scene-play-singletask-task1-v0	19 ± 4	80 ± 0	33 ± 41	**99 ± 1**	**98 ± 3**	**100 ± 0**	**100 ± 0**	**99 ± 1**	5 ± 4	7 ± 10	**100 ± 0**
scene-play-singletask-task2-v0	2 ± 3	9 ± 1	21 ± 7	86 ± 3	79 ± 13	**95 ± 4**	21 ± 18	84 ± 14	0 ± 0	4 ± 3	**95 ± 7**
scene-play-singletask-task3-v0	0 ± 0	21 ± 7	5 ± 7	27 ± 18	30 ± 11	**97 ± 1**	**96 ± 3**	**97 ± 1**	0 ± 0	0 ± 0	**96 ± 6**
scene-play-singletask-task4-v0	1 ± 1	2 ± 0	0 ± 0	9 ± 4	0 ± 0	0 ± 0	0 ± 0	**8 ± 0**	0 ± 0	0 ± 0	1 ± 1
scene-play-singletask-task5-v0	0 ± 0	**3 ± 4**	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
puzzle-3x3-play-singletask-task1-v0	6 ± 3	8 ± 8	**96 ± 3**	55 ± 64	94 ± 3	0 ± 0	**97 ± 1**	**97 ± 0**	86 ± 8	48 ± 25	**100 ± 0**
puzzle-3x3-play-singletask-task2-v0	0 ± 0	2 ± 3	14 ± 14	8 ± 3	36 ± 11	0 ± 0	25 ± 10	17 ± 4	64 ± 3	0 ± 0	**100 ± 0**
puzzle-3x3-play-singletask-task3-v0	1 ± 1	1 ± 1	38 ± 20	32 ± 28	0 ± 0	0 ± 0	9 ± 1	12 ± 0	21 ± 18	7 ± 7	**93 ± 1**
puzzle-3x3-play-singletask-task4-v0 (*)	0 ± 0	1 ± 1	15 ± 1	15 ± 13	4 ± 3	0 ± 0	16 ± 6	16 ± 3	3 ± 4	1 ± 1	**92 ± 6**
puzzle-3x3-play-singletask-task5-v0	2 ± 3	1 ± 1	36 ± 14	3 ± 4	64 ± 6	0 ± 0	23 ± 13	18 ± 3	31 ± 7	0 ± 0	**95 ± 1**
puzzle-4x4-play-singletask-task1-v0	1 ± 1	8 ± 6	19 ± 1	56 ± 3	55 ± 16	26 ± 37	51 ± 1	21 ± 16	21 ± 27	4 ± 3	**62 ± 7**
puzzle-4x4-play-singletask-task2-v0	0 ± 0	7 ± 1	12 ± 0	10 ± 6	3 ± 1	0 ± 0	**39 ± 10**	11 ± 7	6 ± 6	0 ± 0	15 ± 1
puzzle-4x4-play-singletask-task3-v0	0 ± 0	5 ± 4	14 ± 6	21 ± 13	**47 ± 4**	24 ± 3	30 ± 8	22 ± 0	2 ± 3	0 ± 0	38 ± 3
puzzle-4x4-play-singletask-task4-v0 (*)	0 ± 0	8 ± 8	6 ± 3	4 ± 3	18 ± 6	8 ± 6	18 ± 8	17 ± 1	0 ± 0	0 ± 0	**20 ± 0**
puzzle-4x4-play-singletask-task5-v0	0 ± 0	0 ± 0	8 ± 3	3 ± 1	2 ± 0	0 ± 0	**18 ± 0**	8 ± 3	1 ± 1	0 ± 0	7 ± 1
pen-human-v1	14 ± 0	32 ± 6	**69 ± 3**	**72 ± 3**	45 ± 1	35 ± 5	44 ± 0	0 ± 0	2 ± 3	0 ± 0	53 ± 2
pen-cloned-v1	51 ± 2	46 ± 14	47 ± 7	68 ± 1	67 ± 10	52 ± 9	**75 ± 9**	1 ± 1	3 ± 1	0 ± 0	52 ± 3
pen-expert-v1	89 ± 10	96 ± 3	**129 ± 4**	**126 ± 1**	92 ± 3	24 ± 10	95 ± 1	2 ± 3	1 ± 1	0 ± 0	**129 ± 6**
door-human-v1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	**3 ± 4**	0 ± 0	**3 ± 1**	0 ± 0	0 ± 0	0 ± 0	0 ± 0
door-cloned-v1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	**1 ± 1**	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
door-expert-v1	48 ± 13	**104 ± 1**	**105 ± 2**	**103 ± 1**	99 ± 1	**103 ± 0**	**106 ± 0**	**104 ± 0**	0 ± 0	0 ± 0	**103 ± 1**
hammer-human-v1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	**1 ± 1**	0 ± 0	0 ± 0	0 ± 0	0 ± 0	**1 ± 0**
hammer-cloned-v1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	**7 ± 7**	0 ± 0	0 ± 0	0 ± 0	2 ± 1
hammer-expert-v1	89 ± 16	99 ± 1	**134 ± 0**	118 ± 2	84 ± 6	0 ± 0	96 ± 3	125 ± 3	0 ± 0	0 ± 0	126 ± 3
relocate-human-v1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	**3 ± 1**	0 ± 0	2 ± 3	0 ± 0	0 ± 0	0 ± 0	0 ± 0
relocate-cloned-v1	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
relocate-expert-v1	5 ± 3	99 ± 1	**106 ± 1**	**103 ± 1**	98 ± 16	**104 ± 7**	**105 ± 4**	**106 ± 1**	0 ± 0	0 ± 0	**105 ± 4**
visual-antmaze-medium-navigate-singletask-task1-v0 (*)	–	**86 ± 8**	–	–	–	32 ± 3	41 ± 10	**82 ± 3**	–	–	**82 ± 6**
visual-antmaze-medium-navigate-singletask-task2-v0	–	93 ± 7	–	–	–	86 ± 3	91 ± 1	**98 ± 3**	–	–	84 ± 3
visual-antmaze-medium-navigate-singletask-task3-v0	–	90 ± 3	–	–	–	29 ± 41	74 ± 11	**97 ± 4**	–	–	28 ± 6
visual-antmaze-medium-navigate-singletask-task4-v0	–	**95 ± 7**	–	–	–	44 ± 3	72 ± 10	90 ± 3	–	–	78 ± 3
visual-antmaze-medium-navigate-singletask-task5-v0	–	**94 ± 0**	–	–	–	23 ± 3	45 ± 7	**96 ± 3**	–	–	73 ± 7

E.2 Offline-to-Online Evaluation Results

Table 6: **Offline-to-online success-rate results on OGBench.** We report success rates (%) before and after online fine-tuning as offline → online. Results are averaged over evaluation checkpoints and reported as mean ± standard deviation. Bold values indicate the best final online performance.

Task	IQL	QAM	VGF	FQL	**QGD (Ours)**
antmaze-large-navigate-task1	$41 \pm 6 \rightarrow 57 \pm 7$	$1 \pm 1 \rightarrow 0 \pm 0$	$8 \pm 2 \rightarrow 0 \pm 0$	$\mathbf{94 \pm 2 \rightarrow 94 \pm 2}$	$\mathbf{82 \pm 2 \rightarrow 97 \pm 2}$
antmaze-large-navigate-task2	$42 \pm 2 \rightarrow 38 \pm 3$	$13 \pm 3 \rightarrow 10 \pm 4$	$0 \pm 0 \rightarrow 0 \pm 0$	$\mathbf{71 \pm 7 \rightarrow 88 \pm 2}$	$\mathbf{2 \pm 2 \rightarrow 87 \pm 5}$
antmaze-large-navigate-task3	$69 \pm 4 \rightarrow 63 \pm 6$	$68 \pm 9 \rightarrow 74 \pm 6$	$10 \pm 7 \rightarrow 0 \pm 0$	$\mathbf{91 \pm 1 \rightarrow 100 \pm 0}$	$\mathbf{84 \pm 7 \rightarrow 99 \pm 1}$
antmaze-large-navigate-task4	$41 \pm 9 \rightarrow 41 \pm 7$	$29 \pm 1 \rightarrow 29 \pm 4$	$3 \pm 2 \rightarrow 0 \pm 0$	$\mathbf{29 \pm 28 \rightarrow 96 \pm 2}$	$0 \pm 0 \rightarrow 91 \pm 2$
antmaze-large-navigate-task5	$57 \pm 8 \rightarrow 61 \pm 7$	$23 \pm 2 \rightarrow 17 \pm 5$	$5 \pm 5 \rightarrow 0 \pm 0$	$\mathbf{87 \pm 3 \rightarrow 93 \pm 1}$	$\mathbf{71 \pm 7 \rightarrow 95 \pm 2}$
humanoidmaze-medium-navigate-task1	$\mathbf{13 \pm 1 \rightarrow 18 \pm 9}$	$1 \pm 1 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 0 \pm 0$	$1 \pm 1 \rightarrow 0 \pm 0$
humanoidmaze-medium-navigate-task2	$3 \pm 0 \rightarrow 0 \pm 0$	$1 \pm 1 \rightarrow 2 \pm 3$	$1 \pm 1 \rightarrow 0 \pm 0$	$\mathbf{42 \pm 24 \rightarrow 54 \pm 16}$	$0 \pm 0 \rightarrow 0 \pm 0$
humanoidmaze-medium-navigate-task3	–	$\mathbf{9 \pm 3 \rightarrow 11 \pm 2}$	$2 \pm 3 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 0 \pm 0$	$4 \pm 2 \rightarrow 1 \pm 1$
humanoidmaze-medium-navigate-task4	$\mathbf{5 \pm 1 \rightarrow 3 \pm 1}$	$1 \pm 1 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 2 \pm 2$	$0 \pm 0 \rightarrow 0 \pm 0$
humanoidmaze-medium-navigate-task5	$21 \pm 7 \rightarrow 16 \pm 5$	$3 \pm 2 \rightarrow 2 \pm 0$	$1 \pm 1 \rightarrow 0 \pm 0$	$\mathbf{46 \pm 27 \rightarrow 97 \pm 2}$	$1 \pm 1 \rightarrow 1 \pm 1$
cube-double-play-task1	$7 \pm 5 \rightarrow 3 \pm 4$	$31 \pm 6 \rightarrow 0 \pm 0$	$\mathbf{33 \pm 8 \rightarrow 92 \pm 2}$	$0 \pm 0 \rightarrow 0 \pm 0$	$\mathbf{57 \pm 6 \rightarrow 91 \pm 3}$
cube-double-play-task2	$0 \pm 0 \rightarrow 0 \pm 0$	$4 \pm 2 \rightarrow 6 \pm 3$	$29 \pm 2 \rightarrow 55 \pm 8$	$0 \pm 0 \rightarrow 0 \pm 0$	$\mathbf{42 \pm 3 \rightarrow 80 \pm 9}$
cube-double-play-task3	$0 \pm 0 \rightarrow 0 \pm 0$	$3 \pm 1 \rightarrow 0 \pm 0$	$\mathbf{39 \pm 6 \rightarrow 68 \pm 7}$	$0 \pm 0 \rightarrow 0 \pm 0$	$21 \pm 1 \rightarrow 18 \pm 6$
cube-double-play-task4	$0 \pm 0 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 0 \pm 0$	$6 \pm 2 \rightarrow 11 \pm 3$	$0 \pm 0 \rightarrow 0 \pm 0$	$\mathbf{10 \pm 1 \rightarrow 13 \pm 3}$
cube-double-play-task5	$1 \pm 1 \rightarrow 0 \pm 0$	$0 \pm 0 \rightarrow 0 \pm 0$	$\mathbf{43 \pm 8 \rightarrow 60 \pm 6}$	$0 \pm 0 \rightarrow 0 \pm 0$	$43 \pm 4 \rightarrow 50 \pm 5$

F Additional Ablation Studies

This appendix provides additional ablations that complement the main analysis in Section 5.3. We further examine critic-gradient refinement and the sensitivity to downstream drifting affinity normalization. These experiments are supplementary analyses and do not introduce additional components of QGD.

F.1 Critic-Gradient Refinement

We examine whether direct critic gradients provide additional benefits beyond QGD's scalar-Q-based target construction. Table 7 compares full QGD with two gradient-based variants. Across these comparisons, the behavior proposal, critic, policy architecture, training budget, and evaluation protocol are kept fixed. Because the available runs are not seed-matched across variants, the comparisons are descriptive and unpaired.

Gradient-refined targets. This variant replaces QGD's zeroth-order local refinement with direct critic-gradient refinement. Starting from the same behavior-conditioned proposal actions $\{a_m\}_{m=1}^{M}$, we update each target action using one normalized critic-gradient ascent step:

$$a'_m = \mathrm{clip}\left(a_m + \eta \frac{\nabla_a Q_\phi(s, a_m)}{\max(\|\nabla_a Q_\phi(s, a_m)\|, 10^{-8})}, -1, 1\right), \tag{96}$$

where η is the gradient step size. The updated actions $\{a'_m\}$ are directly used as positive targets for drifting. Unlike QGD, this variant does not perform zeroth-order local refinement, candidate evaluation, or critic-based target selection, and therefore isolates the effect of direct critic-gradient optimization.

Gradient + QGD. This variant keeps the QGD target-construction procedure unchanged, but applies one additional normalized critic-gradient ascent step to each selected positive target. The exact dataset action is left unchanged. Thus, the behavior-conditioned proposal, per-anchor zeroth-order refinement, scalar-Q ranking, and positive-target selection are identical to QGD, and only the selected positive targets receive the additional gradient update.

Table 7: **Critic-gradient ablations.** We compare QGD with two critic-gradient variants on four OGBench single-task environments. Values are mean ± standard deviation.

Environment	QGD	Gradient-refined	Gradient + QGD
AntMaze Large (task 1)	$\mathbf{97 \pm 1}$	81 ± 13	96 ± 7
Cube Double (task 5)	$\mathbf{46 \pm 14}$	2 ± 2	23 ± 18
Scene (task 2)	95 ± 7	88 ± 11	$\mathbf{97 \pm 1}$
Puzzle 3×3 (task 4)	86 ± 6	72 ± 15	$\mathbf{89 \pm 15}$
Macro average	**81**	61	76

As shown in Table 7, neither gradient-based variant improves the aggregate performance of full QGD. QGD obtains a macro average of 81, compared with 61 for Gradient-refined targets and 76 for Gradient + QGD. The effect of critic-gradient refinement is also environment-dependent: the gradient-based variants remain competitive on some environments but degrade substantially on others. In particular, both variants underperform QGD on Cube, where the scalar-Q target construction achieves the strongest result.

Together, these comparisons suggest that direct critic action gradients do not provide a consistent aggregate benefit over QGD. Scalar critic evaluations are sufficient to rank, refine, and select useful policy targets, while an additional action-gradient update can alter these targets without reliably improving downstream policy learning.

F.2 Sensitivity to Drifting Affinity Normalization

Finally, we examine whether QGD depends strongly on the particular affinity normalization used by the downstream drifting learner. This ablation leaves QGD target construction unchanged and modifies only the affinity computation in the standard drifting objective described in Appendix A.2. In particular, the positive and negative references, their weights, pairwise distances, affinity logits, and all subsequent drifting computations are kept fixed.

The default drifting affinity in Eq. (28) combines two normalizations: a row-wise softmax over references for each generated sample and a column-wise softmax over generated samples for each reference. We compare this default construction with two one-sided variants. The row-only variant replaces Eq. (28) with

$$A_{iy}^{(R)} = w(y)\ \mathrm{softmax}_y\left(L^{(R)}\right)_{iy}, \tag{97}$$

whereas the column-only variant uses

$$A_{iy}^{(R)} = w(y)\ \mathrm{softmax}_i\left(L^{(R)}\right)_{iy}. \tag{98}$$

Thus, these variants modify only how the pairwise affinity $A_{iy}^{(R)}$ is normalized. The signed positive and negative coefficients, radius-specific drifting fields, field normalization, and final drifting objective remain exactly as defined in Appendix A.2.

Table 8: **Drifting affinity-normalization ablations.** Only the affinity normalization is varied; QGD target construction and the remaining Drifting update are fixed. Values are mean ± standard deviation when multiple runs are available.

Affinity normalization	Cube	Scene	Puzzle	Average
Default (row–column)	$\mathbf{46} \pm 14$	95 ± 7	86 ± 6	76
Row-only	35 ± 12	$\mathbf{99} \pm 1$	$\mathbf{100} \pm 0$	**78**
Column-only	22 ± 3	92 ± 3	96 ± 2	70

As shown in Table 8, the choice of affinity normalization leads to a task-dependent trade-off. The row-only variant obtains the highest macro average of 78, compared with 76 for the default row–column normalization and 70 for the column-only variant. The default normalization performs best

on Cube, however, reaching 46, compared with 35 and 22 for the row-only and column-only variants, respectively. Conversely, the row-only variant performs particularly strongly on Scene and Puzzle.

Across all three affinity-normalization variants, QGD remains effective with the same value-guided target construction. This suggests that the gains of QGD do not depend strongly on the default row–column affinity normalization used by drifting. The value-guided targets remain useful under alternative downstream normalizations, indicating that the precise affinity construction is not central to the QGD formulation.

G Additional Related Work

The main text summarizes prior work at the level of broad interfaces between value information, generative policy learning, sampling-based policy improvement, and drifting. Here, we provide additional context on these directions and clarify mechanism-level differences from QGD.

Offline RL and Expressive Policy Modeling. Offline RL has developed a broad range of mechanisms for controlling distribution shift, including behavior constraints and regularization (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019), conservative value estimation (Kumar et al., 2020), pessimistic model-based objectives such as MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020), stationary-distribution correction as in OptiDICE (Lee et al., 2021), and advantage- or value-weighted policy fitting (Peng et al., 2019; Nair et al., 2021; Wang et al., 2020). Other representative approaches include TD3+BC (Fujimoto & Gu, 2021), IQL (Kostrikov et al., 2022), and subsequent behavior-regularized variants (Xu et al., 2023; Tarasov et al., 2023). Expressive policy classes become particularly relevant when offline datasets contain heterogeneous behaviors or multiple valid action modes (Mandlekar et al., 2022; Rafailov et al., 2024). Accordingly, prior work has modeled decision distributions using sequence models (Janner et al., 2021), trajectory-level diffusion (Janner et al., 2022), conditional generative models (Ajay et al., 2023), hierarchical generative policies (Li et al., 2023), and latent skills (Venkatraman et al., 2024). QGD belongs to this broader class of expressive offline policies, but focuses specifically on how learned scalar values are converted into distributional supervision for a one-step stochastic actor.

Value Guidance in Generative Policies. At a finer level, existing generative offline RL methods differ substantially in how value information interacts with the policy-learning procedure. Advantage-weighted generative policy fitting (Peng et al., 2019; Kang et al., 2023), Diffusion-QL (Wang et al., 2023), and FQL (Park et al., 2025b) incorporate value information through policy fitting or actor optimization. Other approaches constrain policy improvement through diffusion-policy constraints (He et al., 2024), score regularization (Chen et al., 2024a), behavior-distribution regularization (Chen et al., 2024c), pessimistic or entropy-regularized objectives (Zhang et al., 2024), and behavior-constrained generative transport (Zhang et al., 2026).

A complementary set of methods introduces value information into the generative process itself. Critic-gradient-based score learning (Psenka et al., 2024), contrastive or energy-based guidance (Lu et al., 2023), distribution-correction guidance (Mao et al., 2024), continuous Q-function alignment (Chen et al., 2024b), and soft critic-gradient optimization (Fang et al., 2025) modify diffusion generation using critic-derived signals. Energy-Weighted Flow Matching (Zhang et al., 2025), QAM (Li & Levine, 2026), and VGF (Xu et al., 2026) similarly guide expressive generative transport using value- or energy-derived information. These mechanisms differ in whether value information modifies the actor objective, constrains the admissible policy distribution, or alters the generative dynamics themselves.

Another line of work focuses primarily on reducing the inference cost of expressive generative actors. Consistency-based policies (Ding & Jin, 2024) reduce iterative generation through consistency-style objectives, while Shortcut Models (Frans et al., 2025) support generation with flexible integration horizons. OFQL (Nguyen & Yoo, 2026) directly formulates a native one-step generative policy, and MeanFlow-based Q-learning (Wang et al., 2026) learns a direct noise-to-action mapping. These efficiency-oriented approaches primarily determine how an expressive distribution is parameterized or generated, whereas QGD determines which critic-informed target distribution is presented to the generative learner.

Sampling-Based Policy Extraction and Search Amortization. The sampling-based methods summarized in the main text can be further distinguished by the role assigned to the candidate population. CEM (Rubinstein, 1999) and MPPI (Williams et al., 2017) repeatedly update candidate populations according to scalar objective evaluations. MBOP (Argenson & Dulac-Arnold, 2020) and LOOP (Sikchi et al., 2022) instead combine learned models or value functions with trajectory-level planning. Reflect-then-Plan (Jeong et al., 2025) further adapts offline planning according to model uncertainty, while Habitizing Diffusion Planning (Lu et al., 2025) amortizes computationally expensive diffusion planning into faster decision models.

In offline policy extraction, the proposal distribution is typically supplied by a learned behavior or generative policy. SfBC (Chen et al., 2023) samples candidate actions from a generative behavior model and ranks them using a critic, while IDQL (Hansen-Estruch et al., 2023) similarly separates expressive behavior modeling from value-guided action extraction. The role of policy extraction itself has also been systematically studied independently of the underlying offline RL backbone (Park et al., 2024). Policy-Agnostic RL (Mark et al., 2024) makes this separation more explicit by constructing improved action targets through critic-based reranking and local optimization before fitting a downstream policy. Related multimodal planning approaches combine generative proposals with subsequent sampling-based refinement (Mizuta & Leung, 2025).

QGD shares with this family the use of scalar evaluations and candidate refinement, but differs in what the refinement procedure produces. It starts from multiple behavior-conditioned anchors, refines them independently using zeroth-order critic evaluations, and retains a set of high-value actions rather than collapsing the search into a single selected action. These refined actions define positive distributional references, while low-value behavior actions provide additional negative references. The entire procedure is restricted to training and amortized into the final stochastic policy, so neither candidate search nor critic evaluation is required at deployment.

Drifting Models and Extensions. Drifting models (Deng et al., 2026) learn one-step generators by evolving the model-induced distribution toward a supplied target distribution and amortizing this evolution into a stochastic forward mapping. The score-matching interpretation of drifting (Turan et al., 2026) connects its update to score-based generative learning, while its Wasserstein-gradient-flow interpretation (Cao et al., 2026) provides a complementary distributional perspective. Kernel Drifting (Esteban-Casadevall et al., 2026) generalizes the underlying kernel-gradient dynamics, and second-order Drifting (Brown et al., 2026) extends the framework beyond first-order distribution evolution. Drifting has also been extended to receding-horizon control (Foffano et al., 2026) and one-step world modeling (Lu et al., 2026).

Recent work further explores drifting-style objectives for preference and policy optimization. Drifting Preference Optimization (Jiang et al., 2026) uses scalar reward rankings over generated samples to construct detached preference updates. Drift-Based Policy Optimization (Gao et al., 2026) uses a drifting-style one-step policy parameterization for reinforcement-learning-based policy optimization. These directions demonstrate that scalar objectives can interact with drifting-style generators in different ways, but address settings and interfaces different from fixed-dataset offline policy improvement.

QGD specifically uses the critic to construct the empirical reference distribution supplied to drifting. Its refinement begins from behavior-supported action anchors, uses detached scalar Q evaluations rather than critic gradients, preserves multiple local high-value proposals, and explicitly introduces low-value behavior samples as additional negatives. The form of the downstream stop-gradient drifting regression is retained; what changes is the construction of the positive and negative reference samples presented to that learner.

Relation to QGD. The literature above can be distinguished along three complementary axes: where value information enters policy learning, whether candidate search is used during training or deployment, and whether the main contribution concerns the generative parameterization or the target distribution being learned. Existing methods use value information through actor optimization, regularization, generative guidance, candidate ranking, or planning. QGD instead uses detached scalar critic evaluations upstream to construct distributional training supervision, confines candidate search to training, and amortizes the resulting improvement into a stochastic one-step drifting policy.

H LIMITATION

While QGD retains one-step stochastic inference, this efficiency is achieved by moving additional computation to training: target construction requires sampling and evaluating multiple candidates during Q-guided local refinement. The method also relies on the learned critic to rank and refine candidate actions, so errors in value estimation can affect the quality of the resulting training targets. In addition, our current evaluation is concentrated on continuous-control benchmarks, namely OGBench and D4RL, together with visual AntMaze experiments. Extending the evaluation to more diverse visual, high-dimensional, and real-world domains would provide a more complete assessment of the generality of the proposed approach.