DIFFUSION POLICY IMPROVEMENT WITH PROPOSAL-CONDITIONED REFINEMENT FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion and flow policies can model complex behaviors in offline reinforcement learning (RL). However, penalizing their KL divergence from the behavior policy can discourage actions having high critic values with low behavior density. Directly refining behavior proposals may be an alternative, yet Gaussian or deterministic editors limit expressiveness to represent multiple separated modes for the same proposal. In this work, we introduce Proposal-Conditioned Refinement Flows (PReFlow), a policy extraction method combining critic-based proposal selection with a conditional refinement flow. To optimize proposal selection and refinement together, we formulate a KL-regularized objective whose optimum induces a Gibbs policy over final actions under a Gaussian-smoothed behavior prior. The refinement flow can represent multiple high value modes, while a proposal-centered Gaussian reference regulates large action changes. This Gaussian reference further enables us to make use of simulation-free, closed form adjoint matching targets from sampled endpoints and critic gradients, yielding a single velocity regression loss without a backward adjoint solve. On 50 OGBench tasks, PReFlow achieves competitive offline performance and the highest aggregate score among the compared methods after online fine-tuning, reaching 91% after 500K environment steps.

1 INTRODUCTION

Diffusion and flow models provide expressive policy classes for offline reinforcement learning (RL), capable of representing complex, multimodal action distributions (Wang et al., 2023; Hansen-Estruch et al., 2023; Park et al., 2025b). The central challenge is to improve such policies using a learned critic while avoiding out-of-distribution actions, where value estimates can be unreliable (Fujimoto et al., 2019; Kumar et al., 2020). A common approach is behavior regularized policy improvement, which balances critic value against deviation from the behavior policy. QAM (Li & Levine, 2026) optimizes an expressive flow policy under KL regularization. However, this can restrict improvement when high value actions have low behavior density, even if they lie close to likely behavior actions.

Candidate-based policy extraction methods offer another approach, using the critic to select promising actions sampled from the behavior policy (Ghasemipour et al., 2021; Hansen-Estruch et al., 2023). But their output is restricted to the candidates, even when better actions lie nearby. This suggests treating the selected action as a proposal that can be further refined toward higher-value actions.

Residual policies learn additive refinements to base actions (Silver et al., 2018; Johannink et al., 2019; Yuan et al., 2025), where multiple distinct refinements may improve a given state and proposal. However, existing residual policies based on Gaussian (Dong et al., 2026a; Li & Levine, 2026) or deterministic (Mu, 2026; Cheng et al., 2026) refinements offer limited flexibility. SPAR (Zhao et al., 2026) supports multimodal refinements through a conditional VAE, but requires residual reconstruction and latent self-imitation. This leads to the following question:

How can we make local action refinement both expressive and efficient to learn?

To answer this question, we introduce **Proposal-Conditioned Refinement Flows (PReFlow)**, a policy extraction method that models additive action refinements with a conditional flow. A behavior flow generates proposals, and the critic selects the one with the highest value. Conditioned on the state and selected proposal, the flow samples a refinement, which is added to the proposal to produce

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.

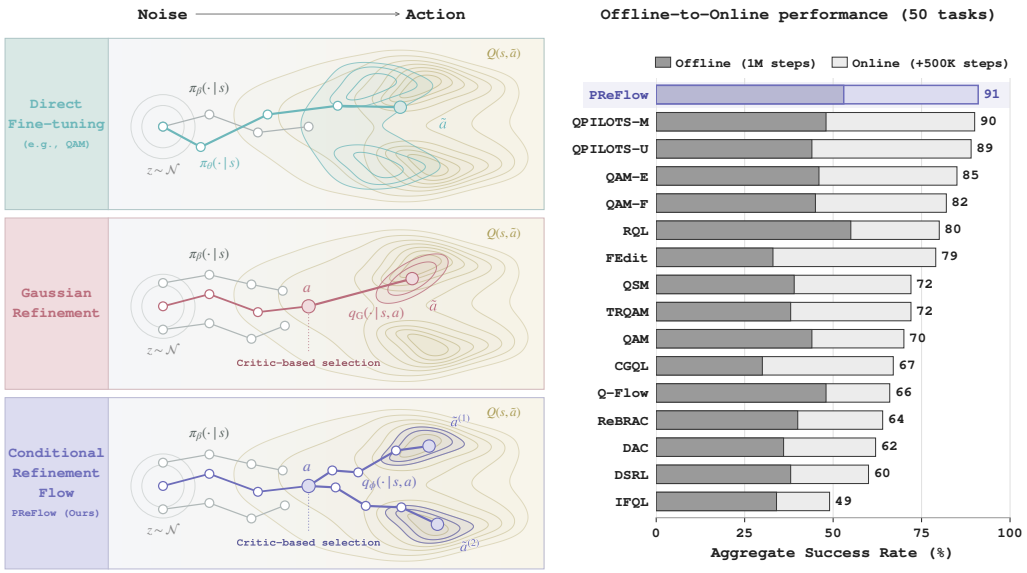


Figure 1: **PReFlow: proposal selection and stochastic refinement.** A behavior flow generates proposals, the critic selects one, and a conditional flow refines it to produce final action $\tilde{a} = a + \Delta a$. The performance panel reports mean success rates across the 50 OGBench tasks (Table 1).

the final action (Figure 1). We regularize the refined-action distribution toward a Gaussian centered at the proposal. This reference discourages large action changes without restricting the refinement distribution to a Gaussian family, allowing distinct high value refinements of the same proposal.

We formulate a joint KL regularized objective for proposal selection and conditional refinement, whose optimum induces a Gibbs policy under a Gaussian-smoothed behavior prior. The Gaussian refinement reference also enables closed form adjoint matching targets (Shin et al., 2026; Havens et al., 2025). Given refinement endpoints from forward rollouts of the current flow, we construct these targets without further simulation and train the conditional flow with a single velocity regression loss.

We evaluate PReFlow on 50 OGBench tasks (Park et al., 2025a), comparing it against a broad range of policy extraction methods for flow and diffusion policies. PReFlow achieves the highest aggregate offline-to-online performance among the compared methods while remaining competitive in offline RL. Our controlled ablations further show gains from expressive flow refinement over Gaussian editing, improvements even with a single proposal, and additional benefits from proposal selection.

2 Preliminaries

We introduce the problem setting and the background underlying our conditional refinement method.

Behavior regularized policy extraction. We consider a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, R, \mu)$, where $\mathcal{S}$ is the state space, $\mathcal{A} = \mathbb{R}^A$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$ is the transition function, $\gamma \in [0, 1)$ is the discount factor, $R : \mathcal{S} \times \mathbb{R}^A \to \mathbb{R}$ is the reward function, and $\mu \in \Delta\mathcal{S}$ is the initial state distribution. The goal of Offline RL is to learn a policy $\pi_\theta : \mathcal{S} \to \Delta\mathcal{A}$ that maximizes the expected discounted return $\mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k)\big]$ from a dataset $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^{|\mathcal{D}|}$ collected by behavior policy π_β. Offline-to-Online (O2O) RL further aims to fine-tune this policy with limited online interactions. Given a learnable critic $Q_\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, we formulate policy extraction with KL regularization toward the behavior policy

$$\max_{\pi} \; \mathbb{E}_{\tilde{a} \sim \pi(\cdot|s)}\big[Q_\psi(s, \tilde{a})\big] - \frac{1}{\tau(s)} \, \mathrm{KL}(\pi \,\|\, \pi_\beta), \tag{1}$$

where $\tau(s) > 0$ controls the critic value relative to the KL penalty (Nair et al., 2020). Optimizing over action distributions gives the Gibbs policy, which favors actions with higher critic values

$$\pi^\star(\tilde{a} \mid s) \propto \pi_\beta(\tilde{a} \mid s) \, e^{\tau(s) Q_\psi(s, \tilde{a})}. \tag{2}$$

Flow matching. To learn the reference π_β in (1), we fit a velocity field f_β to $\mathcal{D}$. This velocity field transforms Gaussian noise into a behavior action through an ordinary differential equation (ODE),

$$\mathrm{d}X_t = f_\beta(X_t, t \mid s)\mathrm{d}t, \quad X_0 \sim \mathcal{N}(0, I_A), \quad X_1 \sim \pi_\beta(\cdot \mid s). \tag{3}$$

To train f_β, we pair a dataset action a at state s with Gaussian noise x_0 and form the linear interpolation $x_t = (1-t)x_0 + ta$. Its velocity provides the regression target in the flow matching loss (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023)

$$\mathcal{L}_{\mathrm{FM}}(\beta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, x_0\sim\mathcal{N},\, (s,a)\sim\mathcal{D}} \left[\| f_\beta(x_t, t \mid s) - (a - x_0) \|_2^2 \right], \tag{4}$$

Q-learning with adjoint matching. QAM (Li & Levine, 2026) learns a flow f_θ to approximate the Gibbs policy (2). Using f_θ as the ODE velocity in (3) defines the policy π_θ and final action $\tilde{a}$,

$$\tilde{a} = \mathrm{ODE}_{f_\theta}(\varepsilon; s) \sim \pi_\theta(\cdot \mid s), \quad \varepsilon \sim \mathcal{N}(0, I_A). \tag{5}$$

Here $\mathrm{ODE}_f(\varepsilon; s)$ denotes the endpoint obtained by integrating the flow f from ε. To train f_θ, QAM uses adjoint matching (AM) (Domingo i Enrich et al., 2025), with f_β fixed as the reference during each policy update. To construct regression targets for this update, AM samples trajectories $\mathbf{X} = \{X_t\}_{t\in[0,1]}$ from a stochastic differential equation (SDE). For a fixed state s, the SDE uses the current flow f_θ and the memoryless noise schedule $\sigma_t = \sqrt{2(1-t)/t}$,

$$\mathrm{d}X_t = \Big(2f_\theta(X_t, t \mid s) - X_t/t \Big)\mathrm{d}t + \sigma_t \, \mathrm{d}B_t, \qquad X_0 \sim \mathcal{N}(0, I_A), \tag{6}$$

where B_t is standard Brownian motion. Along each trajectory, AM computes the lean adjoint $\lambda(\mathbf{X}, t)$ by propagating the endpoint critic gradient backward using the Jacobian of the reference drift,

$$\mathrm{d}\lambda(\mathbf{X}, t) = -\nabla_{X_t} \left[2f_\beta(X_t, t \mid s) - X_t/t \right] \lambda(\mathbf{X}, t)\, \mathrm{d}t, \quad \lambda(\mathbf{X}, 1) = -\tau(s)\nabla_{X_1} Q_\psi(s, X_1). \tag{7}$$

The lean adjoint defines the velocity target $f_\beta - \sigma_t^2\lambda/2$ for f_θ, yielding the regression loss

$$\mathcal{L}_{\mathrm{AM}}(\theta) = \mathbb{E}_{\mathbf{X}} \Big[\int_0^1 \Big\| 2\big(f_\theta(X_t, t \mid s) - f_\beta(X_t, t \mid s) \big)/\sigma_t + \sigma_t \lambda(\mathbf{X}, t) \Big\|_2^2 \mathrm{d}t \Big]. \tag{8}$$

Because (7) depends on the Jacobian of f_β, constructing these targets involves a sequential backward pass of vector–Jacobian products along the entire sampled SDE trajectory.

3 Policy Improvement with Conditional Refinement Flows

In this section, we introduce PReFlow, which learns local value-guided refinements conditioned on proposals from the behavior policy. We motivate this formulation and develop a joint regularized objective for proposal selection and refinement. From this objective, we derive simulation-free adjoint matching targets to train the flow. Proofs and technical details are provided in Appendix A.

3.1 Towards Expressive Conditional Refinement

Behavior regularized flow policies. The goal of policy extraction (1) is to generate final actions from the Gibbs policy $\pi^\star$ in (2). QAM (Li & Levine, 2026) approximates this target by optimizing f_θ with adjoint matching (8), so the learned policy directly generates $\tilde{a} \sim \pi_\theta(\cdot \mid s)$. The KL penalty introduces the behavior-density factor $\pi_\beta(\tilde{a} \mid s)$ into the Gibbs target. Low behavior density can therefore reduce the weight of high value actions even when they lie close to behavior actions. This motivates a conditional approach that regularizes action changes relative to each behavior proposal.

Existing action editors. At state s, the behavior flow f_β trained with the flow matching loss (4) generates a proposal a by integrating the ODE (3). With a fixed, an action editor learns the conditional distribution $\pi^{\mathrm{ref}}(\cdot \mid s, a)$ of an additive refinement Δa. The final action is generated in two stages,

$$a \sim \pi_\beta(\cdot \mid s), \quad \Delta a \sim \pi^{\mathrm{ref}}(\cdot \mid s, a), \quad \tilde{a} = a + \Delta a. \tag{9}$$

Because multiple distinct refinements of the same proposal can yield high critic values at a given state, the conditional policy can benefit from representing multiple modes.

For example, Gaussian-based editors have been proposed to model stochastic refinements (Dong et al., 2026a), while deterministic editors have been developed to map each state and proposal (s, a)

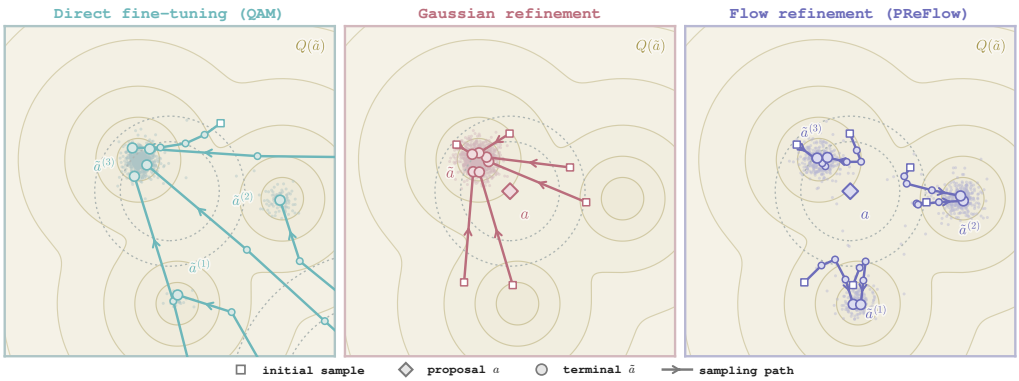


Figure 2: **Toy comparison of policy extraction methods.** Fine-tuning the full flow (QAM) undercovers the two modes farther from the proposal, and Gaussian refinement concentrates on a single mode. PReFlow captures all three modes. Appendix D.4 provides details of the toy experiment.

to a single refinement (Mu, 2026; Cheng et al., 2026). Both parameterizations might restrict the refinement distribution $\pi^{\text{ref}}(\cdot|s,a)$ to be a single mode, limiting its flexibility in capturing distinct refinement modes. A multimodal refinement policy based on a conditional VAE has been proposed to address this limitation (Zhao et al., 2026), but its training combines weighted residual reconstruction with an additional latent self-imitation objective. To retain this expressiveness while simplifying training, we seek a conditional refinement policy learned with a single regression objective.

Proposal-conditioned refinement flows. To this end, we introduce PReFlow, which refines a behavior proposal $a \sim \pi_\beta(\cdot|s)$ using an expressive conditional flow. We train the refinement flow policy $\pi^{\text{ref}}(\cdot|s,a)$ to maximize the expected critic value of the refined action $\tilde{a}$, with KL regularization toward the Gaussian reference $\rho_\delta = \mathcal{N}(0, \delta^2 I)$, where $\delta > 0$ sets the reference refinement scale. Equivalently, the distribution of $\tilde{a} = a + \Delta a$ is regularized toward $\mathcal{N}(a, \delta^2 I)$, therefore the proposal a serves as both an input to the conditional flow and the center of the regularization reference. This reference discourages large action changes without restricting π^{ref} to a Gaussian family, allowing multiple high value refinements of the same proposal as depicted in Fig. 2.

The Gaussian reference also simplifies training by admitting a reference process with closed form intermediate-state distributions and lean adjoints (Domingo i Enrich et al., 2025; Shin et al., 2026). Given refinement endpoints sampled from the current flow, these expressions enable simulation-free construction of regression targets from Gaussian bridge samples and endpoint critic gradients. The resulting single regression objective requires neither a backward adjoint solve nor an additional residual-reconstruction loss. We formalize the conditional objective and use its optimal value to guide proposal selection in the next section, and then derive the regression objective in Section 3.3.

3.2 A Regularized Objective for Conditional Policy Improvement

Starting from policy extraction objective (1), we first derive a refinement objective for Δa conditioned on a fixed behavior proposal a and then use its optimal value to guide proposal selection.

Conditional refinement objective. For state s, we sample a proposal $a \sim \pi_\beta(\cdot \mid s)$ from the behavior flow and hold it fixed during refinement. We optimize $\pi_a(\cdot \mid s)$, the conditional distribution of the refined action $\tilde{a} = a + \Delta a$. To regularize action changes relative to a, we replace the behavior reference in (1) with $\mathcal{N}(a, \delta^2 I)$, as motivated in Section 3.1. For $\tau = \tau(s) > 0$, the objective is

$$\mathbb{E}_{\tilde{a}\sim\pi_a(\cdot|s)}[Q_\psi(s,\tilde{a})] - \frac{1}{\tau}\,\mathrm{KL}\big(\pi_a(\cdot \mid s)\,\|\,\mathcal{N}(a,\delta^2 I)\big)\,. \tag{10}$$

With proposal a fixed, writing $\Delta a = \tilde{a} - a$ gives $\pi_a(a + \Delta a \mid s) = \pi^{\text{ref}}(\Delta a \mid s, a)$. Since the reference for $\tilde{a}$ is $\mathcal{N}(a, \delta^2 I)$, the corresponding reference for Δa is $\rho_\delta = \mathcal{N}(0, \delta^2 I)$. By the translation invariance of KL divergence, we can rewrite the refinement objective in (10) as

$$\mathcal{J}_{s,a}(\pi^{\text{ref}}) = \mathbb{E}_{\Delta a\sim\pi^{\text{ref}}(\cdot|s,a)}[Q_\psi(s,a+\Delta a)] - \tfrac{1}{\tau}\,\mathrm{KL}\big(\pi^{\text{ref}}(\cdot \mid s,a)\,\|\,\rho_\delta\big)\,. \tag{11}$$

Applying the Gibbs solution (2) to (11) gives

$$\pi^{\mathrm{ref},\star}(\Delta a \mid s,a) \propto \exp\left(\tau Q_\psi(s,a+\Delta a) - \frac{\|\Delta a\|^2}{2\delta^2}\right). \tag{12}$$

The target balances critic value against refinement magnitude without directly penalizing low behavior density. It can be multimodal despite its Gaussian reference when the critic favors distinct refinements.

To provide intuition for this optimality, we further define $Z(s,a) = \mathbb{E}_{\Delta a\sim\rho_\delta}[e^{\tau Q_\psi(s,a+\Delta a)}]$, the partition function of the optimal refinement policy. Then, we can rewrite the objective (11) as

$$\mathcal{J}_{s,a}(\pi^{\mathrm{ref}}) = \frac{1}{\tau}\log Z(s,a) - \frac{1}{\tau}\,\mathrm{KL}\big(\pi^{\mathrm{ref}} \,\|\, \pi^{\mathrm{ref},\star}\big). \tag{13}$$

The first term is independent of π^{ref}, and the KL divergence reaches its minimum at $\pi^{\mathrm{ref}} = \pi^{\mathrm{ref},\star}$.

Joint refinement and proposal selection. We have optimized π^{ref} for a fixed proposal a. Since the optimal value of (11) depends on the proposal and balances critic value against the KL penalty, we now optimize proposal selection with refinement. We denote this optimal conditional value by

$$V_\delta(s,a) = \sup_{\pi^{\mathrm{ref}}} \mathcal{J}_{s,a}(\pi^{\mathrm{ref}}) = \frac{1}{\tau}\log Z(s,a). \tag{14}$$

Let $\pi^{\mathrm{base}}(\cdot \mid s)$ denote the distribution of selected proposals. To evaluate selection and refinement jointly, we average $\mathcal{J}_{s,a}(\pi^{\mathrm{ref}})$ over $a \sim \pi^{\mathrm{base}}(\cdot \mid s)$. Since proposals come from π_β, we regularize π^{base} toward it via KL penalty. Using the same coefficient $1/\tau$ for both stages gives the joint objective

$$\mathcal{J}_s(\pi^{\mathrm{base}}, \pi^{\mathrm{ref}}) = \mathbb{E}_{a\sim\pi^{\mathrm{base}}(\cdot|s)}[\mathcal{J}_{s,a}(\pi^{\mathrm{ref}})] - \tfrac{1}{\tau}\,\mathrm{KL}\big(\pi^{\mathrm{base}}(\cdot \mid s) \,\|\, \pi_\beta(\cdot \mid s)\big). \tag{15}$$

With optimal conditional refinement, the first term becomes $\mathbb{E}_{a\sim\pi^{\mathrm{base}}}[V_\delta(s,a)]$. Applying the same Gibbs solution with V_δ in place of Q_ψ therefore gives the optimal proposal-selection policy

$$\pi^{\mathrm{base},\star}(a \mid s) \propto \pi_\beta(a \mid s)\exp(\tau V_\delta(s,a)) = \pi_\beta(a \mid s)Z(s,a). \tag{16}$$

This policy favors proposals according to their value after regularized refinement.

Optimal two-stage policy. We now relate this two-stage policy to the final action target in (2). Using the common coefficient $1/\tau$, the KL chain rule combines the two penalties into a joint KL penalty relative to $\pi_\beta(a \mid s)\rho_\delta(\Delta a)$. This reference first draws a behavior proposal and then adds an independent Gaussian refinement. Its final action distribution is therefore $\bar{\pi}_{\beta,\delta}(\tilde{a} \mid s) = \int \pi_\beta(a \mid s)\rho_\delta(\tilde{a}-a)\,\mathrm{d}a$. This Gaussian-smoothed behavior prior averages Gaussian neighborhoods centered at behavior proposals. The joint optimum reweights the reference by $e^{\tau Q_\psi(s,\tilde{a})}$. Since this weight depends only on the final action, integrating out the proposal gives the following result.

Proposition 3.1 (Final policy under proposal-relative refinement)**.** *Jointly optimizing (15) over all proposal selection and conditional refinement policies yields the final action policy*

$$\pi^\star_\delta(\tilde{a} \mid s) \propto \bar{\pi}_{\beta,\delta}(\tilde{a} \mid s)e^{\tau Q_\psi(s,\tilde{a})}. \tag{17}$$

The two stages of proposal selection and refinement recover the Gibbs policy improvement rule with $\bar{\pi}_{\beta,\delta}$ replacing π_β. Proposals originate from π_β, while δ controls the spread of their Gaussian neighborhoods. Nearby proposals can provide reference density to actions with low behavior density.

Practical proposal selection. Computing $V_\delta(s,a)$ requires integrating over possible refinements, whereas $Q_\psi(s,a)$ is available from a critic evaluation. For a sufficiently smooth critic, the small-δ expansion $V_\delta(s,a) = Q_\psi(s,a) + O(\delta^2)$ motivates using the critic value as an approximation. For K candidates $a_{1:K} \sim \pi_\beta(\cdot \mid s)$, replacing $V_\delta(s,a_i)$ by $Q_\psi(s,a_i)$ in (16) gives finite-temperature selection weights $w_i \propto \exp(\tau Q_\psi(s,a_i))$. We instead use greedy selection as shown in Algorithm 1; Appendix E.3 provides comparison of two methods.

Algorithm 1 PReFlow: Action generation

Input: s, f_β, v_ϕ, Q_ψ, K, δ.
$\varepsilon'_{1:K} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,I)$
$a_{1:K} \leftarrow \mathrm{ODE}_{f_\beta}(\varepsilon'_{1:K}; s)$ ▷ (3)
$I \leftarrow \arg\max_{i\in\{1,\dots,K\}} Q_\psi(s,a_i)$
$a \leftarrow a_I$
$y_0 \sim \mathcal{N}(0,I)$
$y_1 \leftarrow \mathrm{ODE}_{v_\phi}(y_0; s, a)$ ▷ (19)
Output: (a, y_1).

3.3 Simulation-free Training of Flow Refinement

Conditional flow parameterization. To parameterize the refinement policy, we write $\Delta a = \delta y$ and $\tilde{a} = a + \delta y$. Using the Gaussian reference $\rho = \mathcal{N}(0, I)$ for y, we can rewrite (12) as

$$q^\star(y \mid s, a) \propto \rho(y) \exp(\tau Q_\psi(s, a + \delta y)) . \tag{18}$$

We approximate this target with a conditional flow:

$$\frac{\mathrm{d}Y_t}{\mathrm{d}t} = v_\phi(Y_t, t \mid s, a), \qquad Y_0 \sim \mathcal{N}(0, I), \qquad Y_1 \sim q_\phi(\cdot \mid s, a). \tag{19}$$

With the state and proposal fixed throughout the rollout, the endpoint yields the refinement $\Delta a = \delta Y_1$.

Simulation-free target construction. To train this flow, we construct regression targets using the stochastic optimal control (SOC) formulation of Adjoint Matching (Domingo i Enrich et al., 2025). QAM (Li & Levine, 2026) uses the pretrained behavior flow as its reference, so constructing targets requires the sequential backward solve of (7). Our objective instead regularizes the refinement toward the Gaussian reference ρ, while the behavior flow only generates proposals. We therefore use the memoryless linear Gaussian reference flow (Shin et al., 2026), whose velocity field is $v_0(y, t) = (2t - 1)y/d(t)$ with $d(t) = t^2 + (1 - t)^2$. Using v_0 in place of f_θ in the memoryless SDE (6) defines a Gaussian reference process for the normalized refinement y, with terminal distribution ρ and closed form bridges for constructing the targets.

Algorithm 2 PReFlow: Flow Update

Input: $\mathcal{D}$, f_β, v_ϕ, Q_ψ, δ, τ.
Generate endpoints
Sample a replay state $s \sim \mathcal{D}$.
$(a, y_1) \leftarrow$ Algorithm 1, $K = 1$.
$(a, y_1) \leftarrow \mathrm{sg}[(a, y_1)]$ ▷ sg: stop-gradient
Construct the velocity target
$t \sim \mathcal{U}(0, 1)$, $\varepsilon \sim \mathcal{N}(0, I)$
Compute y_t using (20).
$\tilde{a}_1 \leftarrow a + \delta y_1$
Compute v^{tgt} using (21).
Update the refinement flow
Update ϕ by minimizing (22).
Output: v_ϕ.

Since the conditional target is independent of the proposal-selection rule, we draw proposals directly from π_β. Following Adjoint Sampling (Havens et al., 2025), we generate an endpoint $y_1 \sim q_\phi(\cdot \mid s, a)$ with a forward flow rollout and define the refined action as $\tilde{a}_1 = a + \delta y_1$. We then draw $t \sim \mathcal{U}(0, 1)$ and directly sample the intermediate state from the reference process conditioned on y_1 as

$$y_t = t y_1 + (1 - t)\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I). \tag{20}$$

Proposition 3.2 (Simulation-free refinement velocity target)**.** *For an endpoint y_1 and a bridge sample y_t from (20), the adjoint matching velocity target for the Gaussian reference above is*

$$v^{\mathrm{tgt}}(y_t, y_1, t \mid s, a) = \frac{2t - 1}{d(t)} y_t + \frac{\tau \delta (1 - t)}{d(t)} \nabla_{\tilde{a}} Q_\psi(s, \tilde{a}_1). \tag{21}$$

With v_ϕ and v_0 replacing f_θ and f_β, the integrand in (8) becomes $\frac{4}{\sigma_t^2} \| v_\phi - v^{\mathrm{tgt}} \|_2^2$. Following RAM (Bergmeister et al., 2026), we omit this time weight and use unweighted regression on the bridge samples, holding sampled endpoints and targets fixed during the update.

$$\mathcal{L}_{\mathrm{actor}}(\phi) = \mathbb{E}_{\substack{s \sim \mathcal{D},\ a \sim \pi_\beta(\cdot|s),\ y_1 \sim q_\phi(\cdot|s,a) \\ t \sim \mathcal{U}(0,1),\ \varepsilon \sim \mathcal{N}(0,I)}} \Big[\big\| v_\phi(y_t, t \mid s, a) - \mathrm{sg}[v^{\mathrm{tgt}}] \big\|_2^2 \Big], \tag{22}$$

Algorithm 2 summarizes the refinement flow update. The full training algorithm, including critic and behavior flow updates, is provided in Algorithm 3 in Appendix C.3.

4 Related Work

Policy extraction with diffusion and flow policies. Diffusion and flow policies incorporate critic information through end-to-end gradients (Wang et al., 2023), value-weighted fitting (Zhang et al., 2025), score or noise-regression objectives (Psenka et al., 2024; Fang et al., 2025), and energy guidance (Lu et al., 2023; Alles et al., 2025). FQL (Park et al., 2025b) regularizes a separate noise-conditioned one-step actor toward a flow behavior model, while Q-Flow and RQL (Doo et al., 2026; Oberai et al., 2026) learn values over intermediate generative states. QPILOTS and QGF apply critic-gradient guidance during sampling (Ruan et al., 2026; Zhou et al., 2026). PReFlow instead trains its conditional refinement flow with endpoint critic gradients. At inference, critic values guide selection among behavior proposals, while refinement requires no critic-gradient evaluations.

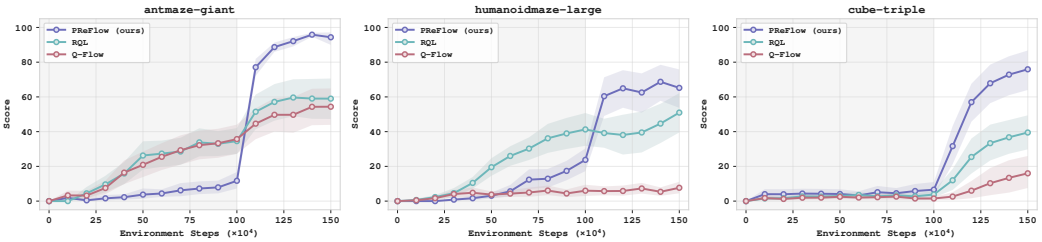


Figure 3: **Learning curves during online fine-tuning.** PReFlow, RQL, and Q-Flow are fine-tuned for 0.5M online environment steps after 1M offline gradient steps. Curves show means over three seeds; shaded regions show pointwise 95% confidence intervals.

Behavior proposals and conditional action refinement. Selection among behavior proposals (Ghasemipour et al., 2021; Hansen-Estruch et al., 2023; Lin et al., 2026) can be combined with local improvement (Fujimoto et al., 2019; Mark et al., 2024; Dong et al., 2026a). Residual policies learn additive modifications to base-policy actions (Silver et al., 2018; Johannink et al., 2019; Yuan et al., 2025). EXPO and QAM-E use Gaussian editors (Dong et al., 2026a; Li & Levine, 2026), whereas DeFlow and Fisher Decorator learn deterministic refinements (Mu, 2026; Cheng et al., 2026). SPAR (Zhao et al., 2026) learns multimodal residuals through a conditional VAE and latent self-imitation, and evaluates flow-based variants. FLAG (Kim et al., 2026) uses local Gaussian policies around flow-generated anchors to supervise policy improvement. PReFlow trains a conditional refinement flow with adjoint matching under proposal-centered KL regularization.

5 Experiments

We evaluate PReFlow in offline and offline-to-online reinforcement learning and examine the effects of refinement policy design, along with computational cost and hyperparameter sensitivity. Detailed experimental settings and hyperparameter settings are provided in Appendix D.

Domains and datasets. We evaluate PReFlow on a total of 50 tasks from the OGBench single-task suite (Park et al., 2025a), spanning antmaze, humanoidmaze, scene, puzzle, and cube domains. We follow QAM's advanced ten domain variants and reward configurations (Li & Levine, 2026).

Baselines. We compare with the policy extraction baselines originally evaluated by Li & Levine (2026): Gaussian policy (ReBRAC (Tarasov et al., 2023)), critic-guided method (QSM (Psenka et al., 2024)), post-processing methods (DSRL, FEdit, and IFQL (Wagenmaker et al., 2025; Li & Levine, 2026)), and adjoint matching methods (QAM, QAM-F, and QAM-E (Li & Levine, 2026)). We also compare with more recent policy extraction methods for flow policies, QPILOTS-M/U (Ruan et al., 2026), Q-Flow (Doo et al., 2026), RQL (Oberai et al., 2026), and TRQAM (Dong et al., 2026b).

5.1 Offline and Offline-to-Online Performance

Following the QAM protocol (Li & Levine, 2026), we train for 1M offline gradient steps and fine-tune for 500K online environment steps. We report success rates at the end of each phase. Domain scores average five tasks, and the overall score averages all 50 tasks. We use eight seeds per task for PReFlow and the reproduced baselines. Task-level results and learning curves are in Appendix E.6.

Table 1 reports offline and offline-to-online performance. PReFlow achieves an aggregate offline score of 53, competitive with RQL and above all evaluated adjoint matching and test-time steering baselines. On `p44`, it exceeds the strongest baseline by about 27 percentage points. PReFlow also outperforms FEdit and IFQL, which use Gaussian editing and selection among behavior samples.

After 500K online environment steps, PReFlow reaches an aggregate score of 91, the highest mean among the compared methods. Figure 3 shows that PReFlow outperforms RQL and Q-Flow after fine-tuning on `ag`,`hl`, and `c3` despite relatively low offline scores. Unlike intermediate value learning methods like RQL and Q-Flow, PReFlow trains its refinement flow directly from critic gradients at the final action. This may help the refiner respond more directly to critic updates from online data, while critic-based proposal selection allows PReFlow to further benefit from these updates.

Method	al (5 tasks)	ag (5 tasks)	hm (5 tasks)	hl (5 tasks)	scene (5 tasks)	p33 (5 tasks)	p44 (5 tasks)	c2 (5 tasks)	c3 (5 tasks)	c4 (5 tasks)	all (50 tasks)
Gaussian											
ReBRAC	[94,95] [97,98] 94→98	[53,60] [69,79] 57→75	[65,74] [52,63] 69→58	[15,19] [18,28] 17→23	[61,69] [98,99] 65→99	[73,84] [100,100] 79→100	[0,0] [0,0] 0→0	[8,10] [96,98] 9→97	[0,1] [9,20] 1→14	[6,11] [79,80] 9→80	[39,41] [63,65] 40→64
Guidance											
QSM	[88,92] [91,96] 90→94	[19,29] [83,91] 24→87	[79,84] [78,81] 82→80	[5,7] [1,3] 6→2	[77,80] [81,83] 78→82	[52,63] [82,90] 57→86	[0,0] [94,100] 0→98	[31,34] [98,99] 33→98	[5,6] [25,28] 6→27	[18,19] [62,71] 19→67	[38,40] [71,73] 39→72
QPILOTS-U	[76,88] [94,98] 82→96	[0,4] [85,94] 1→89	[54,75] [98,100] 63→99	[2,14] [40,72] 8→59	[80,95] [96,100] 89→98	[100,100] [100,100] 100→100	[6,25] [100,100] 16→100	[71,80] [99,100] 75→100	[4,10] [57,74] 7→69	[0,6] [76,80] 3→78	[42,47] [87,90] 44→89
QPILOTS-M	[74,88] [93,98] 82→97	[0,14] [85,95] 10→90	[64,75] [98,100] 71→99	[6,16] [38,70] 12→60	[91,98] [97,100] 95→99	[99,100] [100,100] 100→100	[2,27] [100,100] 17→100	[70,81] [99,100] 75→100	[5,9] [67,75] 7→72	[3,20] [78,80] 10→79	[46,50] [87,91] 48→90
Post-processing											
DSRL	[56,66] [88,91] 61→90	[1,4] [36,45] 3→40	[48,58] [71,82] 53→77	[2,5] [22,36] 3→29	[99,100] [100,100] 99→100	[82,92] [78,95] 87→87	[0,0] [0,0] 0→0	[72,76] [99,99] 74→99	[1,2] [0,0] 1→0	[2,3] [77,78] 2→78	[38,39] [59,61] 38→60
FEdit	[54,62] [95,96] 58→96	[1,3] [83,90] 2→86	[20,23] [72,88] 22→80	[2,3] [5,11] 3→8	[60,65] [98,99] 62→99	[98,100] [100,100] 99→100	[30,37] [100,100] 34→100	[37,43] [100,100] 40→100	[2,3] [34,45] 2→40	[3,7] [79,80] 5→80	[32,33] [78,80] 33→79
IFQL	[32,39] [71,81] 36→76	[0,2] [16,18] 1→17	[85,87] [47,56] 86→51	[21,27] [5,9] 24→7	[83,85] [96,97] 84→96	[100,100] [99,100] 100→100	[0,0] [0,0] 0→0	[10,12] [79,80] 11→79	[0,0] [0,0] 0→0	[1,3] [58,63] 2→61	[34,35] [48,50] 34→49
Intermediate-value learning											
Q-Flow	[94,96] [97,99] 95→98	[32,39] [51,58] 36→54	[86,89] [95,96] 87→96	[7,10] [9,12] 8→10	[96,98] [99,100] 97→99	[100,100] [100,100] 100→100	[0,1] [0,1] 0→0	[34,39] [99,100] 37→99	[2,4] [17,34] 3→25	[11,19] [80,80] 15→80	[47,48] [65,67] 48→66
RQL	[80,84] [88,91] 82→90	[36,45] [54,63] 41→58	[92,99] [98,99] 96→99	[38,43] [48,53] 40→51	[85,89] [97,98] 87→98	[100,100] [100,100] 100→100	[21,43] [100,100] 32→100	[17,20] [89,93] 18→91	[2,4] [33,43] 3→38	[44,53] [79,80] 49→80	[53,56] [80,81] 55→80
Adjoint matching											
QAM	[78,84] [97,98] 81→98	[14,22] [52,54] 18→53	[64,69] [78,88] 67→83	[9,14] [17,24] 11→21	[96,98] [100,100] 97→100	[99,100] [100,100] 100→100	[0,0] [0,8] 0→3	[62,66] [100,100] 64→100	[3,4] [64,76] 3→70	[2,4] [67,74] 3→71	[44,45] [69,71] 44→70
QAM-F	[81,84] [92,94] 83→93	[8,16] [70,74] 12→72	[62,68] [80,90] 65→85	[9,14] [16,23] 12→20	[92,97] [99,100] 95→100	[99,100] [100,100] 99→100	[5,8] [100,100] 6→100	[63,67] [100,100] 65→100	[2,3] [64,73] 3→68	[11,17] [79,80] 14→79	[45,46] [81,82] 45→82
QAM-E	[80,86] [95,97] 83→96	[0,3] [91,97] 1→95	[54,63] [75,90] 59→83	[1,3] [11,22] 2→16	[96,98] [100,100] 97→100	[100,100] [100,100] 100→100	[35,43] [97,100] 39→99	[63,68] [100,100] 65→100	[4,6] [79,79] 5→79	[4,9] [79,80] 6→79	[45,47] [84,86] 46→85
TRQAM	[80,85] [96,97] 82→97	[12,15] [90,94] 14→92	[26,30] [73,79] 28→76	[1,3] [5,9] 2→7	[66,80] [99,100] 72→99	[100,100] [100,100] 100→100	[0,0] [0,0] 0→0	[73,76] [100,100] 75→100	[4,6] [73,78] 5→76	[2,5] [76,77] 3→76	[37,39] [72,73] 38→72
Ours											
PReFlow	[79,85] [98,99] 82→98	[8,15] [91,97] 12→94	[60,69] [91,99] 65→95	[20,28] [58,72] 24→65	[96,97] [98,99] 97→99	[100,100] [100,100] 100→100	[58,74] [100,100] 66→100	[56,62] [100,100] 59→100	[5,8] [74,77] 7→75	[12,17] [80,80] 15→80	[51,54] [90,92] 53→91

Table 1: **OGBench: offline → online success rates (%).** Each cell shows end-of-offline (left) and end-of-online (right) success rates with 95% CIs over seeds in small gray brackets. Best results per domain are highlighted. Per-task results are in Appendix E.6.

5.2 Expressiveness and Efficiency of Conditional Refinement Flow

Refinement policy class on OGBench. We compare no refinement, a learned Gaussian, and PReFlow on four OGBench domains: `al`, `hm`, `scene`, and `c2`. We use a single proposal, and set $\delta = 0.1$ for refinement scale. Both Gaussian and flow refinement policies are trained under the conditional objective in (11). Table 2 shows that both refiners substantially improve over the base policy. The flow achieves higher mean scores than the Gaussian across all four domains, supporting the benefits of expressive conditional refinement.

	Refinement Type		
Domain	Base	Gaussian	Flow
al	2.3±0.8	76.3±7.2	79.0±7.5
hm	2.7±1.0	57.1±8.5	69.2±1.2
scene	3.9±1.5	68.1±3.2	73.1±0.8
c2	2.0±0.0	40.8±5.2	53.9±3.3

Table 2: Offline success rates (%) after 1M steps with $K = 1$.

Computational cost. We next examine the computational cost of learning the flow refinement policy. Figure 4 compares the wall-clock time per full training update with QAM, QAM-E, QPILOTS-U, and QPILOTS-M on GPU, averaged over five flow-step counts: 5, 10, 20, 50, and 100. PReFlow has the lowest mean update time among the compared methods. Its regression target is constructed in closed form without any backward simulation, whereas QAM requires sequential backward adjoint propagation. QPILOTS requires per-step steering with additional estimators to sample actions.

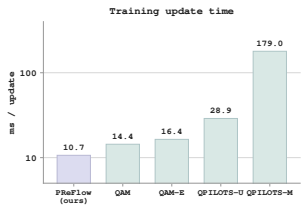


Figure 4: Mean wall-clock time per full training update.

5.3 Effect of Proposal Selection

The policy-class comparison above fixes $K = 1$. We now vary $K \in \{1, 2, 4, 8\}$ while retaining learned refinement in every setting. For $K > 1$, PReFlow selects the behavior proposal with the highest critic value before refinement, while $K = 1$ directly refines a single sampled proposal.

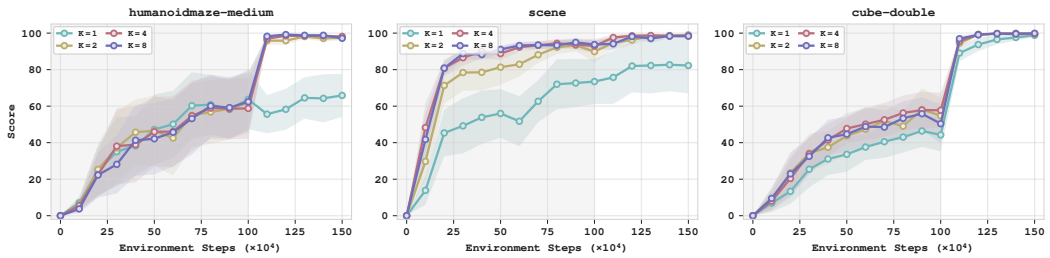


Figure 5: **Effect of proposal selection.** PReFlow with $K \in \{1, 2, 4, 8\}$ behavior proposals on `hm`, `scene`, and `c2`. All settings use the learned flow to refine the proposal with the highest critic value. Curves show means over three seeds; shaded regions show pointwise 95% confidence intervals.

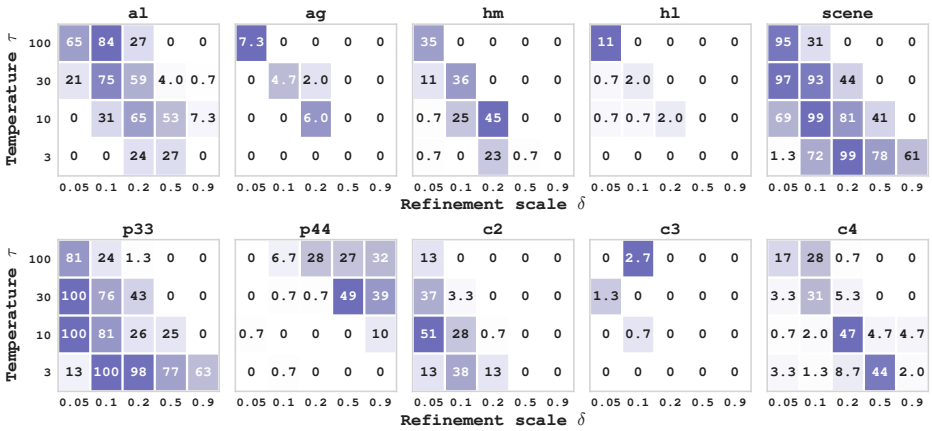


Figure 6: **Sensitivity to the refinement scale and inverse temperature.** Offline performance on OGBench domains across different combinations of the refinement scale δ and inverse temperature τ. Each cell reports the mean performance over three seeds.

Figure 5 presents results on `c2`, `scene`, and `hm`. Overall, proposal selection complements learned refinement, with gains in offline performance and final online success rates already evident with $K = 2$. Full results for all ten OGBench domains are provided in Appendix E.3. We further compare proposal selection with and without refinement in Appendix E.4, showing selection alone improves over the behavior policy but remains far below refinement even with a single proposal.

5.4 Sensitivity to Refinement Scale and Inverse Temperature

We study how the refinement scale δ and inverse temperature τ affect PReFlow's performance. As shown in Figure 6, we find that performance varies with both δ and τ. Under a local linear approximation, the conditional target induces an action refinement with mean $\tau\delta^2\nabla_a Q(s, a)$ and covariance $\delta^2 I$. Both δ and τ therefore control the effective refinement strength.

6 Conclusion

We present PReFlow, a policy extraction method that selects among behavior proposals with the critic and refines the selected proposal with a conditional flow. A proposal-centered Gaussian reference encourages small action changes, while the flow can represent several high-value refinements of the same proposal. The same reference provides closed-form adjoint matching targets from sampled refinement endpoints and critic gradients, allowing simulation-free training of the flow without a backward adjoint solve. On 50 OGBench tasks, PReFlow is competitive in offline RL and attains the highest aggregate success rate after online fine-tuning among the baselines. Our results suggest that expressive policies help not only in modeling multimodal behavior but also in refining each proposal.

Limitation. Proposal selection uses Q_ψ as an approximation to the refinement value V_δ, but proposals with similar critic values can differ in V_δ through the local gradient and curvature of the critic. Efficient estimation of V_δ that captures this dependence is a natural direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript writing and editing, literature searches, and research execution. This included revising passages for clarity and readability, identifying relevant prior work, and supporting implementation tasks such as code drafting, debugging, and refining experimental workflows.

We did not treat AI-generated content as an authoritative source for scientific claims, citations, or experimental conclusions. We checked AI-suggested references against the original sources before citing them or incorporating their findings into the paper. The authors reviewed AI-assisted code and implementation suggestions and tested the resulting code for correctness and consistency with the intended methodology and experimental setup.

REPRODUCIBILITY STATEMENT

The datasets, baselines, experimental setup are described in Section 5 and Appendix D. Additional implementation details, including hyperparameter settings, are provided in Appendix C. The code for reproducing our experiments is available in supplementary materials.

REFERENCES

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Marvin Alles, Nutan Chen, Patrick van der Smagt, and Botond Cseke. FlowQ: Energy-guided flow policies for offline reinforcement learning. *arXiv preprint arXiv:2505.14139*, 2025. URL https://arxiv.org/abs/2505.14139.

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling rl post-training of diffusion and flow-matching models. *arXiv preprint arXiv:2605.10759*, 2026.

Xiaoyuan Cheng, Haoyu Wang, Wenxuan Yuan, Ziyan Wang, Zonghao Chen, Li Zeng, and Zhuo Sun. Fisher decorator: Refining flow policy via a local transport map. *arXiv preprint arXiv:2604.17919*, 2026.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=UWUUVKtKeu.

Carles Domingo i Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 53791–53846, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/852f50969a9e523ec41d26f2f68bd456-Paper-Conference.pdf.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. EXPO: Stable reinforcement learning with expressive policies. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=aFjSjkB6CV.

Yonghoon Dong, Kyungmin Lee, Changyeon Kim, Jaehyuk Kim, and Jinwoo Shin. Trust region q adjoint matching. *arXiv preprint arXiv:2605.27079*, 2026b.

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-flow: Stable and expressive reinforcement learning with flow-based policy. In *International Conference on Machine Learning*, 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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 75743–75763, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/bca7a9a0dd85e2a68420e5cae27eccfb-Paper-Conference.pdf.

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

Abdelghani Ghanem and Mounir Ghogho. Entropy-regularized adjoint matching for offline reinforcement learning. *arXiv preprint arXiv:2605.06156*, 2026.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3682–3691. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ghasemipour21a.html.

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

Aaron J Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Daniel S. Levine, Brandon M Wood, Bin Hu, Brandon Amos, Brian Karrer, Xiang Fu, Guan-Horng Liu, and Ricky T. Q. Chen. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 22204–22237. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/havens25a.html.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019. doi: 10.1109/ICRA.2019.8794127.

Sungha Kim, Gawon Lee, Jusuk Lee, Jonghae Park, H Jin Kim, and Daesol Cho. Flag: Flow policy maxent-rl by latent augmented guidance. *arXiv preprint arXiv:2605.30749*, 2026.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 9010–9041, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0f85efb1e7545dc35a1b5e4d45aaf3c2-Paper-Conference.pdf.

Xuyao Lin, Yixiang Shan, Jinru Duan, Tao Yang, Xinyu Zhao, Runyu Lei, Yiming Zhao, Jiaxin Fan, Zongbao Feng, and Peng Jia. Decoupling policy extraction for offline reinforcement learning. *arXiv preprint arXiv:2608.20909*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

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 Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22825–22855. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/lu23d.html.

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 preprint arXiv:2412.06685*, 2024.

Zhancun Mu. Deflow: Decoupling manifold modeling and value maximization for offline policy extraction. *arXiv preprint arXiv:2601.10471*, 2026.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Aditya Oberai, Seohong Park, and Sergey Levine. Reversal q-learning. *arXiv preprint arXiv:2606.17551*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=M992mjgKzI.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/park25f.html.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41163–41182. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/psenka24a.html.

Yifan Ruan, Chenyang Cao, Andreas Burger, Ali Pesaranghader, Kaveh Kamali, Jaehong Kim, Nandita Vijaykumar, Alan Aspuru-Guzik, Igor Gilitschenski, and Nicholas Rhinehart. Qpilots: Efficient test-time q-steering for flow policies. *arXiv preprint arXiv:2606.14801*, 2026.

Jeongwoo Shin, Dongsoo Shin, Yuchen Zhu, Wei Guo, Yongxin Chen, Joonseok Lee, Jaewoong Choi, and Jaemoo Choi. Efficient adjoint matching for fine-tuning diffusion models. *arXiv preprint arXiv:2605.11480*, 2026.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11592–11620. Curran Associates, Inc., 2023. doi: 10.52202/075280-0511. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/26cce1e512793f2072fd27c391e04652-Paper-Conference.pdf.

Serge Thilges, Onur Celik, Denis Blessing, Emiliyan Gospodinov, and Gerhard Neumann. Scalable maximum entropy reinforcement learning for diffusion policies via adjoint matching. *arXiv preprint arXiv:2606.22630*, 2026.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=jU7AbGq3se.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e5jGTEiJMT.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HA0oLUvuGI.

Jiaxin Zhao, Weihang Pan, Xun Liang, and Binbin Lin. SPAR: Support-preserving action rectification. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=XiaqvkSnWk.

Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

A Proofs and Technical Details

We derive the optimal refinement and selection policies, the adjoint target, and the proposal-selection approximation. We use the notation of the main text, with $\tau = \tau(s) > 0$ and $\delta > 0$.

A.1 Optimal proposal-conditioned policy improvement

Suppose $Q_\psi(s, \cdot)$ is bounded. For the proofs, define

$$Z(s,a) = \mathbb{E}_{\Delta a \sim \rho_\delta}\left[e^{\tau Q_\psi(s, a+\Delta a)}\right], \quad Z_J(s) = \mathbb{E}_{a\sim\pi_\beta}[Z(s,a)]. \tag{23}$$

Both constants are finite and strictly positive. Since the critic is considered bounded, it is sufficient to consider policies with finite KL penalties. Substituting the normalized form of (12) into (11) gives

$$\mathcal{J}_{s,a}(\pi^{\mathrm{ref}}) = \frac{1}{\tau}\log Z(s,a) - \frac{1}{\tau}\,\mathrm{KL}\big(\pi^{\mathrm{ref}} \,\big\|\, \pi^{\mathrm{ref},\star}\big)\,. \tag{24}$$

Thus $\pi^{\mathrm{ref},\star}$ is optimal, with $V_\delta(s,a) = \tau^{-1}\log Z(s,a)$.

Proof of Proposition 3.1. After optimizing conditional refinement, (15) reduces to

$$\mathbb{E}_{a\sim\pi^{\mathrm{base}}}[V_\delta(s,a)] - \tau^{-1}\,\mathrm{KL}(\pi^{\mathrm{base}}\|\pi_\beta). \tag{25}$$

Applying the same Gibbs variational identity gives the optimal selection policy

$$\pi^{\mathrm{base},\star}(a \mid s) = \frac{\pi_\beta(a \mid s)e^{\tau V_\delta(s,a)}}{Z_J(s)} = \frac{\pi_\beta(a \mid s)Z(s,a)}{Z_J(s)}, \tag{26}$$

Substituting this policy and the normalized refinement target into the final-action mixture yields

$$\pi^\star_\delta(\tilde{a} \mid s) = \int \pi^{\mathrm{base},\star}(a \mid s)\pi^{\mathrm{ref},\star}(\tilde{a}-a \mid s,a)\,\mathrm{d}a \quad = \frac{e^{\tau Q_\psi(s,\tilde{a})}}{Z_J(s)}\mathbb{E}_{a\sim\pi_\beta}\left[\rho_\delta(\tilde{a}-a)\right]. \tag{27}$$

Expanding the Gaussian density ρ_δ gives (17), which characterizes the unclipped action distribution. □

A.2 Gaussian reference and adjoint target

As in Section 3.3, write $\Delta a = \delta y$, $\rho = \mathcal{N}(0, I)$. We use the $C = 1$ member of the linear reference family in Shin et al. (2026, Proposition 3.1). With $d(t) = t^2 + (1-t)^2$, its drift and diffusion coefficient are

$$b(y,t) = D(t)y, \qquad D(t) = \frac{2t^2-1}{t\,d(t)}, \qquad \sigma(t)^2 = \frac{2(1-t)}{t}, \quad 0 < t < 1. \tag{28}$$

Its marginals are $\mathcal{N}(0, d(t)I)$, its terminal law is ρ, and its endpoint-conditioned marginal is the bridge in (20). The singular coefficients at $t = 0$ are interpreted through the Gaussian entrance law, see Shin et al. (2026, Appendix B.3) for the reference construction.

Since the terminal law matches the Gaussian prior in (18), the density correction in Shin et al. (2026, Proposition 3.2) vanishes. The terminal cost is therefore $g_{s,a}(y) = -\tau Q_\psi(s, a+\delta y)$. For the controlled SDE $\mathrm{d}Y_t = (b(Y_t,t) + \sigma(t)u_t)\mathrm{d}t + \sigma(t)\mathrm{d}W_t$, the memoryless SOC formulation yields the optimal endpoint law $q^\star$ in (18), under the usual well-posedness and change-of-measure conditions and a realizable terminal tilt (Domingo i Enrich et al., 2025, Section 4 and Appendices C.4 and D).

The reference velocity is $v_0(y,t) = (2t-1)y/d(t)$, and we parameterize the control as

$$u_\phi(y,t \mid s,a) = \frac{2}{\sigma(t)}\big(v_\phi(y,t \mid s,a) - v_0(y,t)\big). \tag{29}$$

This gives controlled drift $2v_\phi - y/t$. Following Adjoint Sampling (Havens et al., 2025), we hold the sampled endpoint fixed during the update and draw intermediate states from (20).

Proof of Proposition 3.2. The reference Jacobian is $\nabla_y b(y,t) = D(t)I$. Thus (7) gives $\dot{\lambda}(t) = -D(t)\lambda(t)$, with $\lambda(1) = -\tau\nabla_y Q_\psi(s, a+\delta y)|_{y=y_1}$. Using $D(t) = \frac{\mathrm{d}}{\mathrm{d}t}\log(d(t)/t)$ and $d(1) = 1$, we obtain

$$\lambda(t) = \exp\left(\int_t^1 D(r)\,\mathrm{d}r\right)\lambda(1) = \frac{t}{d(t)}\lambda(1). \tag{30}$$

The AM control target is $-\sigma(t)\lambda(t)$. Converting it to a velocity through (29) yields

$$\begin{aligned} v^{\mathrm{tgt}}(y_t, y_1, t \mid s, a) &= v_0(y_t, t) - \frac{\sigma(t)^2}{2}\lambda(t) \\ &= \frac{2t-1}{d(t)}y_t + \frac{1-t}{d(t)}\tau\,\nabla_y Q_\psi(s, a+\delta y)\big|_{y=y_1}. \end{aligned} \tag{31}$$

Finally, the chain rule gives $\nabla_y Q_\psi(s, a+\delta y)|_{y=y_1} = \delta\nabla_{\tilde{a}} Q_\psi(s, \tilde{a}_1)$, which establishes (21). □

The control-to-velocity conversion also determines the AM loss weight:

$$\|u_\phi + \sigma(t)\lambda(t)\|^2 = \frac{4}{\sigma(t)^2}\left\|v_\phi - v^{\mathrm{tgt}}\right\|^2. \tag{32}$$

The weight is $4/\sigma(t)^2 = 2t/(1-t)$. Following Bergmeister et al. (2026, Appendix B.2), we omit the weight and sample time uniformly in (22). For a frozen endpoint distribution and finite losses, positive time weights preserve the pointwise conditional-mean minimizer in an unrestricted velocity class, but can change the fit of a finite network.

The idealized Adjoint Sampling characterization requires both population regression stationarity and consistency of the learned dynamics with the reference bridges (Havens et al., 2025, Theorem C.3). Here, consistency requires the ODE velocity to equal the flow-matching velocity associated with its own endpoint distribution and (20). The target derivation alone establishes neither this consistency nor convergence of finite-network training or finite-step sampling to $q^\star$.

A.3 Proposal-selection approximation

We justify replacing $V_\delta(s,a)$ by the critic value in selection weights. Keep τ fixed and suppose $Q_\psi(s,\cdot)$ is bounded and twice continuously differentiable near a. For $F(x) = e^{\tau Q_\psi(s,x)}$ and $Y \sim \rho$, a second-order Taylor expansion gives

$$\mathbb{E}[F(a+\delta Y)] = F(a) + \frac{\delta^2}{2}\mathrm{tr}(\nabla_a^2 F(a)) + o(\delta^2). \tag{33}$$

The linear term vanishes because $\mathbb{E}[Y] = 0$, and the quadratic term uses $\mathbb{E}[YY^\top] = I$. Continuity of the Hessian controls the local remainder, and boundedness of F and Gaussian tails make the contribution outside that neighborhood $o(\delta^2)$. Taking the logarithm and dividing by τ yields

$$V_\delta(s,a) = Q_\psi(s,a) + \frac{\delta^2}{2}\left[\mathrm{tr}(\nabla_a^2 Q_\psi(s,a)) + \tau\|\nabla_a Q_\psi(s,a)\|^2\right] + o(\delta^2). \tag{34}$$

Thus $V_\delta(s,a) = Q_\psi(s,a) + O(\delta^2)$, motivating the leading-order weights $e^{\tau Q_\psi(s,a)}$ for small refinement scales. For i.i.d. behavior proposals, weights $Z(s,a_i) = e^{\tau V_\delta(s,a_i)}$ give a self-normalized importance-sampling approximation to $\pi^{\mathrm{base},\star}$. Since the weights are bounded and positive, the strong law of large numbers gives almost-sure convergence of weighted expectations of bounded test functions to their expectations under $\pi^{\mathrm{base},\star}$ as the number of candidates increases. Replacing V_δ by Q_ψ gives the finite-temperature weights $w_i \propto e^{\tau Q_\psi(s,a_i)}$ used in Section 3.2.

Our implementation selects $I = \arg\max_i Q_\psi(s,a_i)$ before refinement. Table 6 compares greedy and finite-temperature selection empirically.

B Extended Related Work

B.1 Policy extraction with diffusion and flow policies

Q-Flow and RQL differ in how they learn values over intermediate generative states. Q-Flow (Doo et al., 2026) fits an auxiliary value network to critic evaluations at endpoints obtained by integrating

the current flow from intermediate states. Its policy update then uses gradients of this intermediate value network. RQL (Oberai et al., 2026) reconstructs virtual trajectories by running the flow backward from dataset actions and uses multi-step returns to learn values in the expanded MDP. PReFlow evaluates Q_ψ at the final refined action and uses its gradient to construct a velocity target through the analytic reference adjoint. Its actor update requires neither an intermediate value estimator nor value backups across refinement steps.

QPILOTS (Ruan et al., 2026) estimates guidance from clean actions conditioned on an intermediate flow state. QPILOTS-U uses a denoised point estimate, whereas QPILOTS-M trains a meta flow map to produce approximate posterior samples. The auxiliary model in QPILOTS-M therefore supports critic-gradient estimation during sampling. PReFlow's additional flow models refinements conditioned on a behavior proposal. The critic gradient supervises this flow during training, so the refinement rollout requires no critic gradients at inference time.

B.2 Behavior proposals and conditional action refinement

SPAR (Zhao et al., 2026) evaluates flow-based residual policies in addition to its conditional VAE formulation. Its latent self-imitation procedure samples residual candidates from a target policy, weights them using conservative advantage estimates, and fits the residual policy to these weighted samples. Both the weights and sampled residuals are detached during this update. PReFlow specifies a conditional Gibbs target through its proposal-centered KL objective. The critic gradient at a sampled refined action determines a detached velocity target through the analytic adjoint.

FLAG (Kim et al., 2026) uses a local Gaussian policy around each flow-generated anchor. With fixed covariance, the EM update moves each anchor toward the mean of a reweighted local target. Its expressive action marginal arises from mixing these local policies over flow latents. PReFlow's conditional flow can represent multiple refinement modes for the same proposal; its Gaussian reference defines the KL penalty rather than the policy family.

The regularization of refinements also differs across methods. DeFlow constrains the displacement of a deterministic refiner (Mu, 2026), while Fisher Decorator penalizes a local transport map using the behavior score (Cheng et al., 2026). PReFlow's Gaussian KL regularizer balances expected squared displacement against conditional entropy. It discourages large edits without imposing a hard radius on individual refinements or requiring an estimate of the behavior score.

EMaQ and decoupled policy extraction use critic values to select among behavior proposals (Ghasemipour et al., 2021; Lin et al., 2026). PReFlow's joint objective weights proposals by their optimal regularized refinement value, accounting for the value and KL cost of subsequent refinement. At the joint optimum, the unclipped action marginal is a Gibbs policy under the Gaussian-smoothed behavior prior. The implementation uses critic-based argmax selection; this rule does not in general sample the optimal proposal marginal (Appendix A.3).

B.3 Adjoint matching and tractable reference dynamics

QAM applies Adjoint Matching (Domingo i Enrich et al., 2025; Li & Levine, 2026) with a learned behavior flow as the reference. Although its actor update avoids backpropagation through the optimized policy's sampler, target construction requires a backward lean-adjoint solve through the behavior reference. TRQAM (Dong et al., 2026b) adapts the constraint through a path-space trust region, while ME-AM (Ghanem & Ghogho, 2026) modifies entropy regularization and introduces a mixture behavior prior.

EAM (Shin et al., 2026) constructs linear reference dynamics with closed form lean adjoints. To preserve the original reward-tilted pretrained distribution, it corrects the terminal cost for the change of reference; the gradient of this correction depends on the pretrained model's score. PReFlow uses the $C = 1$ member of EAM's reference family. Its terminal law $\rho = \mathcal{N}(0, I)$ already matches the prior in our conditional refinement objective, so the density correction vanishes. The terminal cost is therefore $-\tau Q_\psi(s, a + \delta y)$, while behavior information enters through the proposals.

Adjoint Sampling (Havens et al., 2025) uses reference bridges to separate endpoint generation from the construction of intermediate training samples. AMDP (Thilges et al., 2026) applies reciprocal adjoint matching to maximum-entropy RL. PReFlow uses this training construction for its conditional

refinement objective. Proposals and refinement endpoints are generated by detached forward rollouts. Given an endpoint, the analytic reference bridge and lean adjoint provide intermediate samples and velocity targets without further trajectory simulation or a backward solve. The simulation-free property applies to these operations; generating the endpoints still requires forward rollouts.

Appendix A.2 derives our velocity target and time weighting, and states the consistency conditions underlying the idealized Adjoint Sampling characterization (Havens et al., 2025, Theorem C.3).

C Additional Method Details

C.1 Temperature Adaptation

A fixed τ can lead to different deviations from the reference across domains because the critic gradients have different scales. We use a cheap local approximation of the conditional KL to guide temperature adaptation.

Local KL approximation. For fixed s and a, consider the linear approximation $Q_\psi(s, a+\delta y) \approx Q_\psi(s,a) + \delta g_a^\top y,\ g_a = \nabla_a Q_\psi(s,a)$. With the Gaussian reference $\rho = \mathcal{N}(0, I)$, the corresponding conditional target is $q^\star_{\text{lin}} = \mathcal{N}(\tau\delta g_a, I)$. Writing $d_{\text{act}} = \dim(\mathcal{A})$, its KL per action dimension is $\frac{1}{d_{\text{act}}}\,\text{KL}(q^\star_{\text{lin}}\|\rho) = \frac{\tau^2\delta^2\|g_a\|^2}{2d_{\text{act}}} = \frac{1}{2}\left(\frac{\mathbb{E}_{y\sim q^\star_{\text{lin}}}\|y\|^2}{d_{\text{act}}} - 1\right)$. Under this approximation, estimating the KL requires only squared endpoint norms. We use this expression as a cheap KL proxy for the learned refinement policy.

Temperature update. For endpoints sampled from the current refinement policy, define the normalized second moment, or dispersion, as $\text{disp} = \frac{1}{d_{\text{act}}}\mathbb{E}\|y_1\|^2$. The reference ρ has dispersion one, regardless of the action dimension or chunk length. For the linearized target, a dispersion target $c \geq 1$ therefore corresponds to a conditional KL of $(c-1)/2$ per action dimension. We adjust τ to keep the measured dispersion close to c using $\mathcal{L}_\tau = \log\tau\,(\text{sg}[\text{disp}] - c)$. We initialize τ to 10 on `al`, `ag`, and `p44`, and to 30 on `c4`. The temperature is updated at every gradient step.

C.2 Implementation details

Critic and behavior-flow updates. We follow QAM (Li & Levine, 2026) for the critic architecture, ensemble aggregation, Bellman backup construction, and target-network updates. The behavior-flow initialization and training schedule during offline and online learning also follow QAM. Critic values and action gradients use the same ensemble aggregation rules as QAM. We train the critic with the same expectile regression objective as RQL (Oberai et al., 2026), using the domain-specific κ values in Appendix D.2. Training proposals for the refinement-flow update are drawn directly from π_β, without proposal selection, as in (22). For each TD target, we also sample one proposal from π_β and refine it with q_ϕ. The K-candidate selection rule in Algorithm 1 is used for action generation.

Action selection and clipping. A single batched behavior-flow rollout generates the K candidate proposals. Our benchmark policy chooses the candidate with the highest critic value and samples a refinement conditioned on that proposal. When $K = 1$, this reduces to refining the only available proposal. The resulting action $\tilde{a} = a + \delta y$ is clipped to the environment's action bounds.

C.3 Training procedure

Algorithm 3 summarizes one joint update of the critic, behavior flow, and refinement flow. Following QAM (Li & Levine, 2026), we use ten critics $\{Q_{\psi^j}\}_{j=1}^{10}$ and their target networks $\{Q_{\bar{\psi}^j}\}_{j=1}^{10}$. Let $\bar{Q}_{\text{mean}}$ and $\bar{Q}_{\text{std}}$ denote the mean and standard deviation of the target ensemble. The pessimistic target value is

$$\bar{Q}(s,a) = \bar{Q}_{\text{mean}}(s,a) - \rho_Q \bar{Q}_{\text{std}}(s,a), \tag{35}$$

where ρ_Q corresponds to QAM's pessimism coefficient ρ. The critic Q_ψ used for proposal selection and refinement denotes the mean of the critic ensemble.

Algorithm 3 PReFlow: Joint training update

Input: Minibatch $\mathcal{B}$ of replay samples (s, a, s', r, m), behavior flow f_β, refinement flow v_ϕ, critic parameters ψ, target parameters $\bar{\psi}$, discount γ, backup horizon h, pessimism ρ_Q, target update rate λ, refinement scale δ, inverse temperature τ, and expectile κ.

Critic update
Generate $\tilde{a}'$ at each s' using Algorithm 1 with $K = 1$.
Clip $\tilde{a}'$ to the action bounds.
Update each critic ψ^j, $j = 1, \ldots, 10$, by minimizing
$\mathcal{L}(\psi^j) = \ell_\kappa\big(\mathrm{sg}\big[r + \gamma^h m\, \bar{Q}(s', \tilde{a}')\big] - Q_{\psi^j}(s, a)\big)$.

Behavior-flow update
Update β using the flow-matching loss in (4) on (s, a) from $\mathcal{B}$.

Refinement-flow update
Update ϕ using Algorithm 2 on the states in $\mathcal{B}$. ▷ Training proposals use $K = 1$
if automatic temperature adaptation is enabled **then**
  Update $\log \tau$ as described in Appendix C.1, using the endpoints from the refinement update.
end if

Target-network update
$\bar{\psi} \leftarrow (1 - \lambda)\bar{\psi} + \lambda\psi$.

Output: Updated $\beta, \phi, \psi, \bar{\psi}$, and τ.

We retain our expectile extension of QAM's squared TD-error loss: $\ell_\kappa(e) = \big|\kappa - \mathbf{1}_{\{e<0\}}\big| e^2$, where e is the target minus the prediction. The default $\kappa = 0.5$ recovers squared-error regression up to a constant factor; we use $\kappa = 0.7$ on `c2` and `c4`. For an h-step backup, r denotes the accumulated discounted reward, s' the bootstrap state, and m the bootstrap mask. The target therefore uses the discount $\gamma^h m$. Losses below are averaged over the minibatch, and target parameters are initialized as $\bar{\psi} \leftarrow \psi$.

D Experimental Details

D.1 Benchmark and evaluation protocol

Our implementation builds on the official QAM codebase (Li & Levine, 2026), and we follow its experimental protocol on the OGBench single-task benchmark (Park et al., 2025a). Unless otherwise stated, we use QAM's default architecture and optimization settings for the shared components, including its domain-specific discount factors and critic pessimism coefficients; see Appendices E and F of Li & Levine (2026).

We evaluate ten OGBench domains, with five tasks per domain (50 tasks total): `antmaze-large` (`al`), `antmaze-giant` (`ag`), `humanoidmaze-medium` (`hm`), `humanoidmaze-large` (`hl`), `scene`, `puzzle-3x3` (`p33`), `puzzle-4x4` (`p44`), `cube-double` (`c2`), `cube-triple` (`c3`), and `cube-quadruple` (`c4`).

Following Li & Levine (2026), we use the default `navigate` datasets for `al`, `ag`, `hm`, and `hl`, and the `play` datasets for the manipulation domains. We use the 100M-transition datasets for `p44` and `c4`, and sparse rewards for `scene`, `p33`, and `p44`. Action chunking uses $h = 5$ for manipulation and $h = 1$ for navigation.

The benchmark protocol consists of 1M offline gradient steps followed by 500K online environment steps. For our runs, we evaluate every 50K steps using 50 episodes per task and seed, and report success rates at the end of each phase. Domain scores average five tasks, and the overall score averages all 50 tasks. For the main benchmark, PReFlow and our reproductions of TRQAM, Q-Flow, and RQL use eight seeds per task. Result sources and seed counts for the other baselines are given in Appendix D.3. We use three seeds for ablation studies and additional analyses, unless otherwise stated. The separate toy and ablation protocols are specified in Appendices D.4, D.5, and D.6.

Hardware. We ran our experiments on NVIDIA GPUs, including RTX 3090, RTX A6000, RTX 4090, and RTX 5090 models. Computational cost comparisons were measured on a single NVIDIA RTX A6000 GPU, with the timing protocol detailed in Appendix D.7.

Domain	τ	δ	K	Notes
al	auto (c=2.5)	0.1	1	–
ag	auto (c=1.5)	0.1	1	δ_{online}=0.2
hm	30	0.1	4	–
hl	100	0.05	4	–
scene	3	0.1	4	sparse
p33	3	0.1	4	sparse
p44	auto (c=2.0)	0.9	1	sparse, $100M$
c2	3	0.1	8	$\kappa = 0.7$
c3	30	0.1	1	–
c4	auto (c=15)	0.1	4	$\kappa = 0.7$, $100M$

Table 3: **Per-domain PReFlow settings.** "auto (c)" denotes the temperature adaptation of Appendix C.1 with target c.

D.2 PReFlow hyperparameters

For our proposal-conditioned refinement velocity network, we use a four-layer MLP with 512 hidden units per layer, where the state and proposal are concatenated as input. Both the behavior and refinement flows use 10 Euler integration steps.

Table 3 reports the per-domain PReFlow settings. We search over fixed inverse temperatures $\tau \in \{3, 10, 30, 100\}$ and dispersion targets $c \in \{1.5, 2.0, 2.5, 5.0, 15.0\}$ for automatic temperature adaptation (Appendix C.1). We also search over refinement scales $\delta \in \{0.05, 0.1, 0.2, 0.5, 0.9\}$ and numbers of behavior proposals $K \in \{1, 2, 4, 8\}$. We evaluate each candidate configuration with three seeds per task, and select the configuration that maximizes the sum of the success rates at the end of offline training and online fine-tuning. We then fix this configuration for all five tasks in the domain and report results averaged over eight seeds per task.

We use QAM's squared TD-error objective (Li & Levine, 2026) as the default for critic learning. Following RQL (Oberai et al., 2026), we also consider an asymmetric version that gives different weights to predictions below and above their TD targets. This expectile regression loss is controlled by κ: our default $\kappa = 0.5$ recovers QAM's loss up to a constant factor, while $\kappa > 0.5$ gives more weight to predictions below their targets. We use $\kappa = 0.7$ only on `c2` and `c4`.

TRQAM (Dong et al., 2026b) increases its KL budget at the offline-to-online transition on `ag`. Motivated by this choice, we test increasing PReFlow's refinement scale δ from 0.1 during offline training to 0.2 during online fine-tuning. We apply this adjustment only to `ag`. Both this adjustment and the expectile choices above are treated as domain-specific hyperparameter settings.

Sensitivity sweep. For Figure 6, we vary δ and fixed τ with $K = 1$ and $\kappa = 0.5$, without temperature adaptation. We evaluate one task per domain, using task 1 for navigation and task 2 for manipulation. For `c4`, all tested configurations yielded zero success on task 2, so we instead use task 1 to illustrate performance differences across hyperparameter settings. Each configuration is trained for 1M offline gradient steps without online fine-tuning. Each cell reports the mean success rate over three seeds at the 1M-step checkpoint.

D.3 Baselines

For ReBRAC, FBRAC, BAM, FQL, FAWAC, CGQL and its variants, DAC, QSM, DSRL, FEdit, IFQL, and QAM and its variants, we use the released results from QAM (Li & Levine, 2026), with 12 seeds per task. Table 1 reports a subset of these methods; the full task-level tables additionally include FBRAC, BAM, FQL, FAWAC, CGQL-M, and CGQL-L. We refer to Appendices E and F of Li & Levine (2026) for their implementations and hyperparameter settings. We re-run TRQAM, Q-Flow, and RQL using the protocol in Appendix D.1, with eight seeds per task. For QPILOTS-M and QPILOTS-U, we use the authors' released results with eight seeds per task, as detailed below.

TRQAM. TRQAM (Dong et al., 2026b) extends QAM with a path-space KL constraint relative to the behavior flow. It adjusts a trust-region parameter in the sampling dynamics through projected dual updates, using the estimated KL along sampled trajectories to track a prescribed budget ϵ_{KL}.

We use the authors' reported budget for ϵ_{KL} except `hm` and `hl`. Using the published budget of $\epsilon_{\mathrm{KL}} = 0.5$ left the KL constraint unsatisfied under our protocol, with the dual variable reaching its optimization boundary.

Following the authors' tuning protocol, we sweep $\epsilon_{\mathrm{KL}} \in \{1.0, 2.0, 4.0\}$ on tasks 1 and 4 using two seeds per configuration. We select $\epsilon_{\mathrm{KL}} = 1.0$ for both domains and fix it across all five tasks. We do not use BC pre-training.

Q-Flow. Q-Flow (Doo et al., 2026) learns a value function over intermediate flow states. For a sampled intermediate state, it runs the policy to the endpoint and uses the terminal critic value as a regression target. The policy is then trained by adding the gradient of this intermediate value function to the standard flow-matching target, avoiding backpropagation through the full flow rollout.

We use the authors' released implementation. The original paper averages offline evaluations at 800K, 900K, and 1M gradient steps. We instead report the evaluation at 1M steps to match the other methods in our comparison.

RQL. RQL (Oberai et al., 2026) treats individual flow-refinement steps as actions in an expanded MDP. It reverses flows from dataset actions to construct virtual trajectories for off-policy learning, and reduces the effective value-learning horizon introduced by the refinement steps. This allows it to train the full flow policy without backpropagating through the entire sampling process.

We extend the implementation to support online data collection and the streaming OGBench datasets, and use the authors' reported per-domain hyperparameters.

QPILOTS. QPILOTS (Ruan et al., 2026) steers a behavior flow at inference time by computing critic guidance from clean-action estimates at each flow step. QPILOTS-U uses a Tweedie point estimate, while QPILOTS-M uses approximate samples from the conditional endpoint distribution generated by an auxiliary meta flow map.

We use the authors' released results for both variants, based on eight seeds per task and the hyperparameter settings reported in their paper.

D.4 Toy example: conditional multimodality

This example illustrates the conditional refinement policies in Section 3.1 and Figure 2. We adapt the three-peak continuous bandit of QVPO (Ding et al., 2024) and use its reward as a fixed critic, omitting the state argument:

$$Q(\tilde{a}) = \sum_{i=1}^{3} \frac{w}{2\pi\sigma^2} \exp\left(-\frac{\|\tilde{a} - \mu_i\|^2}{2\sigma^2}\right). \tag{36}$$

We use $w = 1.5$, $\mu_1 = (-1.35, 0.65)$, $\mu_2 = (-0.65, 1.35)$, $\mu_3 = (-1.61, 1.61)$, and $\sigma = 0.2$ in place of the original 0.1. We fix the proposal $a = (-1.40, 1.40)$, the refinement scale $\delta = 0.5$, and the inverse temperature $\tau = 2$. Writing $\Delta a = \delta y$, the conditional objective in (11) gives the target

$$q^\star(y) \propto \mathcal{N}(y; 0, I)\, e^{\tau Q(a+\delta y)}. \tag{37}$$

We compare flow refinement, full-covariance Gaussian refinement, and QAM (Li & Levine, 2026) on the same problem. The two refinement policies use the same conditional objective, critic, proposal, reference, δ, and τ. All three methods are trained for 100K steps using Adam with global-norm gradient clipping at 1.0. The initial learning rate is $\eta_0 = 5 \times 10^{-4}$, reduced to $\eta_0/10$ at 60% of training and $\eta_0/30$ at 90%. The flow and Gaussian use batch sizes of 4096 and 32,768, respectively. The flow uses a 3×256 MLP with a zero-initialized output layer and 32 Euler integration steps. The Gaussian is trained using reparameterization gradients of the same conditional objective. For QAM, the behavior policy is set to the reference law $\pi_\beta = \mathcal{N}(a, \delta^2 I)$. We realize it as a rectified flow from $\mathcal{N}(0, I)$ with a closed form velocity field. QAM fine-tunes this flow in action space by adjoint

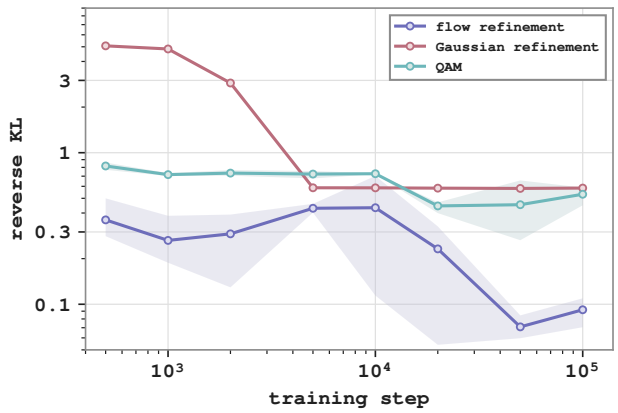


Figure 7: **KL divergence to the target on the toy problem.** Lines show means over 3 seeds, and shaded regions show one standard deviation. Both axes are logarithmic.

matching using (6)-(8) with the same critic and τ, and uses the same batch size, network, and number of discretization steps as the flow refinement. Its target density is proportional to $\pi_\beta(\tilde{a})e^{\tau Q(\tilde{a})}$ and corresponds to $q^\star$ under $\tilde{a} = a + \delta y$.

Figure 7 tracks the KL divergence from each learned distribution q to the target during training, $D_{\mathrm{KL}}(q \,\|\, q^\star) = \mathbb{E}_{y\sim q}\big[\log q(y) - \log q^\star(y)\big]$, which we estimate with 100,000 samples from q. Flow refinement attains the lowest mean KL at every evaluated step and ends below 0.1.

D.5 Refinement policy class on OGBench

We describe the refinement policy comparison in Table 2 of Sec. 5.2. We use a learned diagonal Gaussian policy, $q_\phi(y \mid s, a) = \mathcal{N}(\mu_\phi(s,a), \mathrm{diag}(\sigma^2_\phi(s,a)))$. The learned Gaussian uses the same state and proposal inputs and MLP hidden-layer sizes as the refinement flow, and is trained to maximize

$$\tau\, \mathbb{E}_{y\sim q_\phi(\cdot|s,a)}\left[Q_\psi(s, a+\delta y)\right] - \mathrm{KL}(q_\phi(\cdot \mid s,a) \| \mathcal{N}(0, I))\,. \tag{38}$$

We also include a variant without refinement ($\delta = 0$). The learned Gaussian and flow both use $\delta = 0.1$; the no-refinement variant returns the behavior proposal directly. All variants use $K = 1$. We evaluate on `al`, `hm`, `scene`, and `c2`, with five tasks each and three seeds per task. Both refiners use fixed inverse temperatures of 100, 30, 3, and 3 on `al`, `hm`, `scene`, and `c2`, respectively. We use $\kappa = 0.7$ on `c2` and $\kappa = 0.5$ elsewhere. We report offline success rates after 1M gradient steps, evaluated over 50 episodes per task and seed.

D.6 Proposal selection

We compare three rules for selecting among K behavior proposals: uniform sampling, sampling with probabilities proportional to $\exp(\tau Q_\psi(s, a_i))$, and selecting the proposal with the highest critic value. The selected proposal is then refined using the learned refinement flow. For softmax selection, we use the domain-specific τ values in Table 3. Each setting is evaluated over three seeds, with each seed covering all five tasks in a domain, as reported in Table 6.

D.7 Timing methodology

Figure 4 reports mean full-training-update times on a single NVIDIA RTX A6000 GPU, averaged over five flow-step counts: 5, 10, 20, 50, and 100.

Table 5 provides a separate timing breakdown at 10 Euler steps. We use `c2` input shapes: 37-dimensional observations and 5-dimensional actions with chunk length $h = 5$ (25-dimensional action chunks), batch size 256, and 10 Euler steps per flow rollout. Each method runs in a separate subprocess. After five warm-up calls, we time 30 calls with device synchronization via `jax.block_until_ready` and report the median in milliseconds.

Method	al 5 tasks	hm 5 tasks	scene 5 tasks	c2 5 tasks
	[43,52]	[5,8]	[12,15]	[2,4]
SPAR anchor	48	7	14	3
	[55,64]	[6,9]	[26,29]	[2,4]
SPAR	60	8	27	3
	[79,85]	[60,69]	[96,97]	[56,62]
PReFlow	82	65	97	59

Table 4: **Comparison with SPAR.** Offline success rates (%) averaged over eight seeds and five tasks per domain, with 95% confidence intervals in small gray brackets above the means. Best results per domain are highlighted. SPAR is evaluated after 1M gradient steps of critic and anchor training followed by 1M steps of residual training.

Method	Actor grad.	Critic grad.	Full step
QAM	3.40	2.13	8.22
QAM-E	5.08	2.71	9.97
PReFlow (ours)	2.65	2.43	7.18
QPILOTS-U	0.84	6.99	10.93
QPILOTS-M	0.91	49.69	51.20

Table 5: **Training-update cost on c2 (ms).** Median times over 30 calls after warm-up, measured on the same GPU. Actor and critic columns measure loss-gradient computation without optimizer updates; the full step includes all training updates. The lowest time in each column is highlighted.

E Additional Experimental Results

E.1 Comparison with SPAR

SPAR (Zhao et al., 2026) also refines an anchor action with a residual policy. It trains a conditional VAE over the residual by advantage-weighted reconstruction and latent self-imitation. We port the authors' implementation to the QAM codebase and keep SPAR's own critic, anchor, and action-selection rule.

SPAR first trains an IQL critic and a tanh-Gaussian anchor for 1M gradient steps, and then trains the residual for 1M steps with both networks frozen. At evaluation, it executes the best of ten residual candidates only when their robust critic value exceeds that of the anchor. Network sizes, action chunking, discount factors, and the evaluation protocol match PReFlow. We tune the anchor (Gaussian or behavior flow), the advantage filter (hard or soft), the guide weight $\lambda_g \in \{0.5, 1, 2\}$, and the ensemble penalty $\lambda_u \in \{0, 0.5, 1\}$ on `al`, `hm`, `scene`, and `c2` with three seeds per task.

The paper's main configuration (Gaussian anchor, hard filter, $\lambda_g = 1$, $\lambda_u = 0.5$) gives the highest average, so we run it with eight seeds per task. Table 4 reports offline success rates after 1M gradient steps of residual training. SPAR improves on its anchor on `al` and `scene`, but it remains below PReFlow in all four domains.

E.2 Computational cost

Table 5 provides the timing breakdown at 10 Euler steps. The timing methodology is described in Appendix D.7.

E.3 Effect of Proposal Selection

Figure 9 provides the full learning curves for the experiment in Section 5.3 across all ten OGBench domains.

Table 6 shows that critic-guided selection improves performance over uniform selection on `c2` and `scene`, while the improvement on `hm` is limited. Argmax achieves the highest mean success rate in

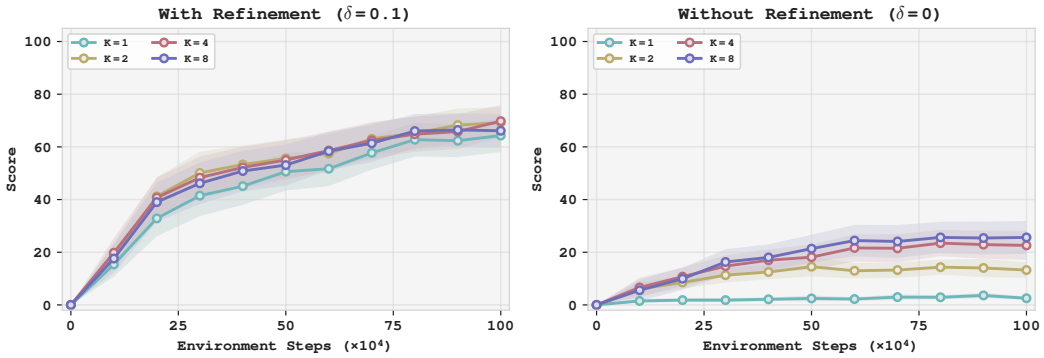


Figure 8: **Proposal selection with and without refinement.** Offline success rates average on `c2`, `hm`, `scene`, `al` for $K \in \{1, 2, 4, 8\}$, with refinement ($\delta = 0.1$, left) and without refinement ($\delta = 0$, right). Without refinement, the selected proposal is executed directly. Curves show means over three seeds; shaded regions show pointwise 95% confidence intervals.

four of the six domain-K settings and remains competitive in the other two. Based on these results, we use argmax selection in our benchmark experiments. Appendix A.3 distinguishes greedy selection from the finite-temperature policy derived from the joint objective.

		$K = 4$			$K = 8$		
Domain	$K = 1$	**Uniform**	**Softmax**	**Argmax**	**Uniform**	**Softmax**	**Argmax**
hm ($\tau = 30$)	69.2±1.2	68.8±2.6	70.1±3.1	70.0±2.9	68.7±0.8	66.0±3.3	69.2±3.4
scene ($\tau = 3$)	73.1±0.8	74.0±1.7	89.5±0.8	94.9±1.0	74.1±0.8	90.7±1.7	92.8±0.6
c2 ($\tau = 3$)	53.9±3.3	50.3±6.9	61.3±1.0	63.3±4.2	49.6±8.0	60.1±3.9	58.7±4.1

Table 6: **Effect of the selection rule.** Offline success rates (%) after 1M gradient steps. Entries report mean ± standard deviation over three seeds, with each seed's score averaged over five tasks. The listed τ values apply to softmax selection. Best results are highlighted among the three rules for each domain and K.

E.4 Proposal selection without refinement

To separate the contributions of proposal selection and refinement, we repeat the K sweep of Section 5.3 with refinement disabled ($\delta = 0$), so that the selected behavior proposal is executed directly. All other settings follow Section 5.3. Figure 8 reports offline success rates averaged over four domains (`al`, `hm`, `scene`, `c2`). Without refinement, $K = 1$ reduces to the behavior policy and scores near zero. Selecting among more proposals helps, but remains far below refinement with a single proposal. Refinement therefore accounts for most of the improvement, while proposal selection complements it.

E.5 Domain-specific hyperparameter choices

The hyperparameter settings are given in Appendix D.2.

Critic expectile. With all other hyperparameters fixed, increasing κ from 0.5 to 0.7 improves offline performance by 4.9 and 5.9 percentage points on `c2` and `c4`, respectively. This change does not consistently improve performance on the other domains. We hypothesize that increased critic overestimation limits its benefit on those domains.

Refinement scale during online fine-tuning. Increasing the refinement scale δ from 0.1 during offline training to 0.2 during online fine-tuning improves online performance on `ag`, but does not yield significant gains on other domains.

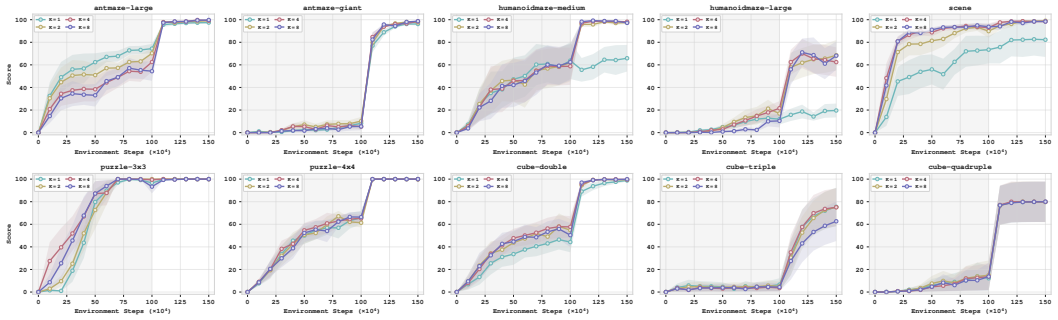


Figure 9: **Effect of proposal selection across all ten OGBench domains.** PReFlow with $K \in \{1, 2, 4, 8\}$ behavior proposals. All settings use the learned refinement flow to refine the proposal with the highest critic value. Curves show means over three seeds; shaded regions show pointwise 95% confidence intervals.

E.6 Full task-level results

Tables 7 and 8 report success rates on all 50 tasks at the end of offline training and after online fine-tuning, respectively. PReFlow, TRQAM, Q-Flow, and RQL use eight seeds per task under our common protocol. For the baselines evaluated by Li & Levine (2026), we use their released results with 12 seeds per task. Each task cell reports the mean success rate with a 95% confidence interval over seeds. Aggregate rows report the mean across the five tasks in each domain. Figure 10 provides the task-level learning curves. QPILOTS-M/U are included in the domain-level comparison in Table 1, but not in these task-level tables; their result sources are given in Appendix D.3.

E.7 Full training curves

Figure 10 presents the training curves of PReFlow on all 50 tasks, together with the curves of the baselines evaluated by Li & Levine (2026), plotted from their released results. Online fine-tuning begins after 1M offline gradient steps and continues for 500K environment steps.

Task	ReBRAC	FBRAC	BAM	FQL	FAWAC	CGQL	CGQL-M	CGQL-L	DAC	QSM	DSRL	FEdit	IFQL	QAM	QAM-F	QAM-E	TRQAM	Q-Flow	RQL	PReFlow
al																				
task1	98	0	90	93	6	63	56	39	88	87	62	67	41	77	85	87	85	96	80	79
task2	88	0	60	86	1	73	49	51	71	78	75	67	14	80	64	81	62	92	78	78
task3	98	12	94	59	40	91	90	79	98	99	81	65	54	89	96	94	93	98	94	90
task4	94	0	85	54	18	78	78	76	91	91	18	28	25	69	81	70	80	92	80	79
task5	95	0	88	86	22	75	80	78	92	96	69	63	45	89	87	83	91	96	76	82
agg.	94	2	84	76	17	76	71	65	88	90	61	58	36	81	83	83	82	95	82	82
ag																				
task1	36	0	3	0	0	2	0	0	53	62	0	0	0	7	11	0	2	11	15	5
task2	73	0	0	0	0	0	0	4	0	1	0	0	0	0	0	0	8	42	56	22
task3	13	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	4	47	0
task4	74	0	0	0	0	0	0	0	0	0	12	10	2	34	5	0	7	50	23	7
task5	89	0	1	0	0	0	22	11	25	58	2	0	2	49	43	6	52	72	63	24
agg.	57	0	1	0	0	0	4	3	16	24	3	2	1	18	12	1	14	36	41	12
hm																				
task1	38	26	49	34	18	30	8	55	87	88	49	0	86	40	32	27	15	90	99	59
task2	91	78	69	95	44	78	99	93	96	96	91	39	92	97	98	99	38	96	100	98
task3	83	28	75	96	20	78	0	62	92	95	36	0	93	96	95	68	28	95	100	68
task4	37	3	22	14	1	23	0	2	43	31	0	0	60	3	0	0	9	56	84	0
task5	96	59	83	99	36	89	100	98	99	98	90	68	98	99	99	99	51	99	99	99
agg.	69	39	60	68	24	60	42	62	83	82	53	22	86	67	65	59	28	87	96	65
h1																				
task1	36	0	5	8	0	2	0	2	0	10	14	8	36	6	8	6	0	12	76	6
task2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	2
task3	32	0	6	19	1	11	4	18	0	16	1	3	55	16	19	4	6	18	37	29
task4	10	0	6	9	0	5	8	4	0	2	0	1	2	16	12	0	3	6	46	47
task5	7	0	11	9	0	5	16	9	1	2	2	1	29	19	19	0	1	5	40	35
agg.	17	0	5	9	0	5	6	6	0	6	3	3	24	11	12	2	2	8	40	24
scene																				
task1	98	66	100	99	62	79	100	100	100	100	100	95	93	100	100	100	100	100	100	100
task2	90	80	99	71	15	88	99	98	99	78	100	97	63	98	99	98	100	100	67	100
task3	51	41	99	97	14	23	92	91	79	97	97	61	71	100	95	100	90	96	94	97
task4	60	49	96	92	71	0	6	86	8	92	100	35	98	100	93	100	52	99	98	97
task5	27	17	96	27	0	0	74	66	53	25	100	24	95	87	88	88	22	92	76	89
agg.	65	50	98	78	38	38	74	88	68	78	99	62	84	97	95	97	72	97	87	97
p33																				
task1	99	1	26	99	8	87	100	98	98	89	83	100	100	98	98	100	100	100	100	100
task2	77	0	78	83	2	55	100	92	66	85	83	100	100	100	100	100	100	100	100	100
task3	85	0	78	83	1	24	100	87	54	18	83	98	100	100	100	100	100	100	100	100
task4	62	0	92	84	1	25	100	85	72	87	83	97	100	100	100	100	100	100	100	100
task5	70	1	9	6	1	47	100	88	51	9	100	100	100	100	100	100	100	100	100	100
agg.	79	0	56	70	3	48	100	90	68	57	87	99	100	100	99	100	100	100	100	100
p44																				
task1	0	29	0	17	0	56	0	0	0	0	0	71	0	0	20	85	0	0	74	80
task2	0	13	0	1	0	12	0	0	0	0	1	13	1	0	5	7	0	0	11	50
task3	0	16	0	3	0	41	0	0	0	0	0	39	0	0	3	56	0	1	34	82
task4	0	6	0	2	0	11	0	0	0	0	0	14	0	0	4	29	0	0	23	65
task5	0	13	0	2	0	3	0	0	0	0	0	32	0	0	2	19	0	0	19	54
agg.	0	15	0	5	0	24	0	0	0	0	0	34	0	0	6	39	0	0	32	66
c2																				
task1	29	0	84	46	8	55	50	62	36	80	90	77	16	85	89	96	56	43	88	88
task2	6	0	49	46	0	39	46	52	35	29	87	28	12	79	84	77	86	32	24	55
task3	2	0	38	42	0	44	50	52	31	33	85	44	10	54	59	55	82	20	12	59
task4	1	0	8	13	0	13	11	18	16	5	32	14	4	22	18	22	35	10	4	32
task5	4	0	56	50	0	42	48	41	56	16	76	39	11	82	81	83	75	64	10	61
agg.	9	0	47	46	2	38	41	45	35	33	74	40	11	64	65	65	75	37	18	59
c3																				
task1	4	2	14	15	0	39	40	40	24	26	7	11	2	14	11	16	19	12	10	32
task2	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	2	2	0	2	0
task3	0	0	3	1	0	1	0	0	0	0	0	0	0	2	2	2	4	2	0	0
task4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
task5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
agg.	1	0	3	3	0	8	8	8	5	6	1	2	0	3	3	5	5	3	3	7
c4																				
task1	35	0	0	11	0	0	2	2	12	90	9	19	8	13	67	32	16	50	72	59
task2	6	0	0	0	0	0	0	0	1	0	1	2	0	0	1	0	0	23	85	1
task3	3	0	0	0	0	0	0	0	0	0	1	2	2	1	3	0	0	1	70	15
task4	0	0	0	0	0	0	0	0	0	4	0	1	0	0	0	0	0	0	19	0
task5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
agg.	9	0	0	2	0	0	1	0	3	19	2	5	2	3	14	6	3	15	49	15

Table 7: **Per-task offline success rates (%) after 1M gradient steps.** Each cell shows the mean over seeds, with a 95% confidence interval in small gray brackets above it. `agg.` rows report the mean across the five tasks. Best results in each row are highlighted, including ties.

Task	ReBRAC	FBRAC	BAM	FQL	FAWAC	CGQL	CGQL-M	CGQL-L	DAC	QSM	DSRL	FEdit	IFQL	QAM	QAM-F	QAM-E	TRQAM	Q-Flow	RQL	PReFlow
a1																				
task1	98 [90,95]	95 [48,94]	97 [62,73]	100 [95,98]	2 [0,1]	99 [80,86]	90 [41,50]	84 [34,42]	96 [71,82]	97 [61,84]	96 [63,76]	97 [87,91]	65 [70,78]	98 [92,96]	96 [74,82]	98 [88,92]	98 [87,91]	99 [91,96]	86 [75,89]	99 [93,96]
task2	93 [99,100]	72 [74,100]	68 [98,100]	97 [100,100]	1 [56,66]	83 [97,100]	46 [85,98]	38 [89,97]	77 [99,100]	73 [100,100]	70 [98,99]	89 [100,100]	74 [84,89]	94 [100,100]	78 [98,100]	90 [99,100]	89 [99,100]	94 [100,100]	82 [99,100]	95 [100,100]
task3	100 [98,99]	91 [77,100]	99 [92,96]	100 [99,100]	61 [18,22]	98 [94,98]	96 [88,90]	94 [83,90]	100 [97,99]	100 [99,100]	98 [87,92]	100 [95,97]	87 [80,86]	100 [96,99]	99 [92,97]	100 [94,97]	100 [98,100]	100 [98,100]	100 [92,96]	100 [96,99]
task4	98 [98,100]	92 [96,99]	94 [95,98]	100 [99,100]	20 [18,28]	96 [95,98]	89 [87,92]	87 [88,93]	98 [98,100]	99 [99,100]	90 [92,96]	96 [96,98]	83 [58,83]	97 [99,100]	94 [96,98]	96 [95,99]	99 [98,100]	99 [97,100]	94 [85,90]	98 [98,100]
task5	99 [97,98]	98 [83,95]	96 [90,92]	100 [99,99]	23 [20,23]	96 [94,95]	90 [81,83]	90 [77,80]	99 [93,95]	99 [91,96]	94 [88,91]	97 [95,96]	73 [71,81]	99 [97,98]	97 [92,94]	97 [95,97]	97 [96,97]	99 [97,99]	88 [88,91]	99 [98,99]
agg.	98	89	91	99	21	94	82	79	94	94	90	96	76	98	93	96	97	98	90	98
ag																				
task1	80 [94,98]	64 [58,96]	25 [0,0]	89 [96,99]	0 [0,0]	92 [91,94]	2 [80,90]	34 [70,79]	54 [94,96]	39 [88,99]	25 [5,28]	90 [76,95]	0 [3,9]	14 [72,79]	70 [88,93]	89 [98,99]	90 [95,98]	33 [44,73]	23 [94,98]	96 [95,99]
task2	96 [10,28]	79 [55,94]	0 [0,0]	97 [70,97]	0 [0,0]	92 [40,66]	88 [6,25]	74 [0,3]	95 [90,96]	98 [97,99]	16 [0,1]	87 [53,77]	6 [0,0]	76 [2,5]	90 [7,11]	98 [86,97]	96 [75,88]	59 [4,8]	96 [40,60]	97 [64,95]
task3	19 [62,96]	77 [73,96]	0 [0,35]	87 [84,99]	0 [0,0]	55 [93,96]	14 [83,95]	2 [83,90]	94 [94,98]	98 [100,100]	1 [51,77]	65 [89,96]	0 [0,4]	3 [72,77]	9 [89,93]	93 [96,99]	82 [91,95]	6 [75,84]	50 [8,45]	83 [96,98]
task4	81 [95,98]	88 [92,97]	14 [0,42]	94 [96,99]	0 [0,0]	94 [91,95]	90 [90,95]	87 [84,89]	96 [98,100]	100 [98,100]	66 [90,96]	93 [96,98]	2 [75,82]	75 [95,98]	91 [98,99]	98 [94,98]	93 [97,100]	79 [93,97]	26 [95,99]	97 [98,100]
task5	97 [69,79]	95 [72,88]	17 [5,18]	98 [88,97]	0 [0,0]	93 [82,88]	93 [55,60]	87 [54,59]	99 [84,91]	99 [83,91]	93 [36,45]	97 [83,90]	79 [16,18]	97 [52,54]	98 [70,74]	96 [91,97]	98 [90,94]	95 [51,58]	97 [54,63]	99 [91,97]
agg.	75	81	11	93	0	85	57	57	87	87	40	86	17	53	72	95	92	54	58	94
hm																				
task1	14 [96,99]	27 [89,93]	53 [91,95]	74 [98,100]	22 [52,62]	44 [85,92]	80 [99,100]	82 [94,98]	87 [96,98]	84 [97,99]	68 [69,91]	75 [58,99]	39 [46,66]	41 [98,99]	63 [100,100]	73 [99,100]	98 [91,97]	98 [98,100]	84 [99,100]	84 [92,100]
task2	98 [50,99]	91 [35,53]	93 [84,89]	99 [96,100]	57 [20,26]	88 [91,94]	100 [69,100]	96 [96,98]	97 [98,100]	98 [98,100]	81 [76,92]	81 [56,98]	56 [32,54]	99 [83,100]	100 [92,100]	100 [75,100]	94 [66,81]	99 [98,100]	100 [98,100]	97 [99,100]
task3	74 [0,9]	44 [1,10]	86 [22,28]	98 [46,81]	24 [2,5]	92 [33,46]	88 [73,99]	97 [95,44]	99 [47,64]	99 [42,26]	85 [88,68]	79 [44,89]	42 [44,58]	94 [66,93]	97 [58,76]	91 [37,42]	98 [40,59]	74 [79,80]	99 [97,99]	100 [91,100]
task4	4 [98,100]	6 [63,89]	25 [95,99]	65 [95,100]	4 [42,57]	40 [91,94]	90 [100,100]	40 [98,100]	56 [98,100]	19 [98,100]	53 [94,98]	68 [100,100]	51 [57,80]	82 [98,100]	57 [99,100]	60 [100,100]	44 [95,99]	83 [99,100]	98 [99,100]	97 [99,100]
task5	99 [52,63]	78 [46,53]	97 [70,72]	98 [82,91]	50 [29,34]	92 [70,73]	100 [86,97]	99 [81,84]	99 [86,90]	99 [78,81]	96 [71,82]	100 [72,88]	69 [47,56]	99 [78,88]	100 [80,90]	100 [75,90]	97 [72,79]	100 [95,96]	100 [98,99]	100 [91,99]
agg.	58	49	71	87	31	71	92	83	88	80	77	80	51	83	85	83	76	96	99	95
h1																				
task1	44 [0,2]	4 [0,0]	0 [0,1]	11 [0,0]	0 [0,0]	1 [0,0]	17 [0,0]	9 [0,0]	25 [0,0]	0 [0,0]	45 [13,37]	19 [0,0]	1 [0,1]	4 [0,0]	13 [0,0]	7 [0,12]	4 [0,1]	17 [0,0]	60 [2,8]	83 [2,18]
task2	1 [28,58]	0 [12,18]	0 [10,47]	0 [36,48]	0 [0,1]	0 [9,15]	0 [51,79]	0 [17,23]	0 [65,78]	0 [7,13]	25 [21,58]	0 [1,17]	0 [16,34]	0 [28,52]	0 [34,53]	4 [33,76]	0 [13,26]	0 [18,27]	4 [89,94]	10 [65,99]
task3	44 [12,30]	16 [0,0]	27 [10,29]	41 [10,21]	0 [0,0]	12 [5,9]	67 [32,55]	20 [5,9]	72 [31,53]	10 [0,0]	39 [5,38]	8 [3,17]	25 [4,10]	41 [19,42]	44 [14,30]	57 [0,18]	19 [2,11]	24 [4,9]	92 [42,58]	88 [34,88]
task4	20 [1,14]	0 [0,0]	20 [13,13]	15 [6,16]	0 [0,0]	7 [3,6]	44 [24,56]	7 [4,10]	43 [30,52]	0 [0,0]	20 [2,32]	10 [0,6]	7 [2,6]	31 [22,32]	22 [11,28]	6 [0,23]	6 [1,10]	7 [3,6]	50 [42,52]	65 [77,84]
task5	7 [18,28]	0 [3,5]	22 [10,19]	10 [11,18]	0 [0,0]	4 [4,6]	40 [28,39]	7 [7,10]	41 [32,40]	0 [1,3]	16 [22,36]	3 [5,11]	4 [5,9]	27 [17,24]	20 [16,23]	8 [11,22]	4 [5,9]	5 [9,12]	47 [48,53]	80 [58,72]
agg.	23	4	14	16	0	5	34	8	36	2	29	8	7	21	20	16	7	10	51	65
scene																				
task1	100 [100,100]	100 [100,100]	100 [84,100]	100 [99,100]	100 [99,100]	100 [99,100]	100 [98,100]	100 [96,99]	100 [99,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [99,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]
task2	100 [98,100]	100 [100,100]	94 [100,100]	100 [99,100]	100 [50,86]	99 [90,96]	99 [97,99]	98 [73,81]	100 [99,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [99,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [99,100]	100 [100,100]	100 [100,100]
task3	99 [98,100]	100 [98,100]	100 [97,99]	100 [99,100]	69 [99,100]	93 [68,75]	98 [86,93]	77 [54,66]	100 [91,97]	100 [96,99]	100 [100,100]	100 [97,99]	100 [97,99]	100 [99,100]	100 [98,100]	100 [98,100]	100 [97,100]	100 [97,100]	100 [84,93]	100 [97,99]
task4	99 [98,98]	99 [89,98]	98 [96,100]	100 [52,64]	100 [92,97]	72 [51,68]	90 [69,80]	60 [1,4]	94 [81,95]	98 [8,15]	100 [99,100]	98 [93,97]	98 [82,88]	100 [100,100]	99 [99,100]	99 [98,100]	99 [96,100]	98 [97,100]	89 [98,100]	98 [94,98]
task5	96 [98,99]	94 [98,100]	100 [96,100]	58 [90,93]	95 [89,96]	60 [83,87]	75 [91,94]	2 [66,69]	93 [96,98]	12 [81,83]	100 [100,100]	95 [98,99]	85 [96,97]	100 [100,100]	99 [99,100]	100 [100,100]	98 [99,100]	98 [99,100]	99 [97,98]	96 [98,99]
agg.	99	99	98	92	93	85	92	67	97	82	100	99	96	100	100	100	99	99	98	99
p33																				
task1	100 [100,100]	100 [100,100]	100 [75,100]	100 [100,100]	100 [2,5]	100 [100,100]	100 [100,100]	100 [100,100]	91 [52,54]	99 [98,100]	83 [58,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]
task2	100 [100,100]	92 [100,100]	92 [100,100]	100 [100,100]	4 [0,2]	100 [100,100]	100 [99,100]	100 [100,100]	42 [10,21]	99 [57,84]	83 [58,100]	100 [100,100]	100 [99,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]
task3	100 [100,100]	100 [100,100]	100 [59,100]	100 [100,100]	1 [0,1]	100 [100,100]	100 [88,91]	100 [100,100]	100 [15,29]	72 [77,90]	83 [58,100]	100 [100,100]	100 [97,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]
task4	100 [100,100]	100 [62,100]	84 [10,59]	100 [100,100]	1 [0,2]	100 [100,100]	88 [100,100]	100 [100,100]	21 [14,28]	84 [65,88]	83 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]
task5	100 [100,100]	84 [90,100]	34 [74,88]	100 [100,100]	1 [21,21]	100 [100,100]	100 [97,98]	100 [100,100]	21 [35,42]	77 [82,90]	100 [78,95]	100 [100,100]	100 [99,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]
agg.	100	95	82	100	21	100	97	100	38	86	87	100	100	100	100	100	100	100	100	100
p44																				
task1	0 [0,0]	100 [100,100]	0 [0,0]	100 [100,100]	0 [0,1]	100 [75,100]	42 [1,38]	39 [12,60]	0 [0,0]	100 [98,100]	0 [0,0]	100 [100,100]	0 [0,0]	0 [0,25]	100 [100,100]	100 [100,100]	0 [0,0]	0 [0,1]	100 [100,100]	100 [100,100]
task2	0 [0,0]	100 [58,100]	0 [0,0]	100 [100,100]	0 [0,1]	92 [100,100]	17 [15,63]	35 [0,33]	0 [0,0]	99 [99,100]	0 [0,0]	100 [100,100]	0 [0,0]	8 [0,1]	100 [100,100]	100 [84,100]	0 [0,0]	0 [0,1]	100 [100,100]	100 [100,100]
task3	0 [0,0]	83 [75,100]	0 [0,0]	100 [100,100]	0 [0,0]	37 [74,100]	14 [0,35]	1 [5,44]	0 [0,0]	100 [98,100]	0 [0,0]	100 [100,100]	0 [0,0]	0 [0,0]	100 [100,100]	94 [100,100]	0 [0,0]	0 [0,1]	100 [100,100]	100 [100,100]
task4	0 [0,0]	92 [58,100]	0 [0,0]	100 [100,100]	0 [0,1]	91 [75,100]	16 [10,56]	23 [0,30]	0 [0,0]	99 [74,100]	0 [0,0]	100 [100,100]	0 [0,0]	0 [0,25]	100 [100,100]	100 [100,100]	0 [0,1]	0 [0,1]	100 [100,100]	100 [100,100]
task5	0 [0,0]	83 [85,98]	0 [0,0]	100 [100,100]	0 [0,1]	92 [88,100]	32 [20,38]	13 [16,34]	0 [0,0]	91 [94,100]	0 [0,0]	100 [100,100]	0 [0,0]	8 [0,8]	100 [100,100]	100 [97,100]	0 [0,0]	0 [0,1]	100 [100,100]	100 [100,100]
agg.	0	92	0	100	0	95	29	25	0	98	0	100	0	3	100	99	0	0	100	100
c2																				
task1	99 [98,99]	97 [93,98]	100 [99,100]	100 [99,100]	10 [1,4]	100 [98,100]	100 [97,99]	100 [99,100]	100 [100,100]	100 [99,100]	100 [97,100]	100 [100,100]	100 [95,99]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [99,100]	99 [97,99]	100 [99,100]
task2	99 [97,99]	96 [85,98]	99 [98,100]	100 [99,100]	2 [1,2]	99 [99,100]	98 [97,100]	100 [98,100]	100 [99,100]	99 [99,100]	98 [99,100]	100 [100,100]	97 [96,98]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	98 [99,100]	98 [96,99]	100 [100,100]
task3	98 [92,98]	96 [56,80]	99 [95,99]	100 [100,100]	2 [0,0]	99 [77,93]	98 [90,96]	99 [95,99]	100 [96,98]	99 [98,100]	100 [96,99]	100 [99,100]	97 [4,8]	100 [99,100]	100 [99,100]	100 [100,100]	100 [99,100]	98 [98,99]	100 [51,68]	100 [100,100]
task4	95 [92,96]	68 [26,56]	97 [95,99]	100 [100,100]	0 [8,23]	87 [98,100]	93 [96,99]	97 [98,100]	97 [98,100]	99 [92,96]	98 [98,100]	100 [100,100]	6 [95,99]	100 [100,100]	100 [100,100]	100 [99,100]	100 [100,100]	98 [98,100]	60 [98,100]	100 [100,100]
task5	94 [96,98]	41 [75,83]	97 [98,99]	100 [100,100]	15 [4,8]	99 [95,98]	98 [97,98]	99 [99,99]	99 [99,99]	94 [98,99]	99 [99,99]	100 [100,100]	97 [79,80]	100 [100,100]	100 [100,100]	100 [100,100]	100 [100,100]	100 [99,100]	99 [89,93]	100 [100,100]
agg.	97	79	98	100	6	97	98	99	99	98	99	100	79	100	100	100	100	99	91	100
c3																				
task1	61 [0,24]	20 [0,2]	83 [40,91]	100 [38,84]	0 [0,0]	100 [0,0]	100 [0,1]	99 [0,2]	100 [4,13]	99 [8,20]	1 [0,0]	85 [0,1]	0 [0,0]	100 [4,39]	100 [42,86]	100 [97,99]	100 [78,96]	47 [8,52]	90 [15,32]	100 [94,97]
task2	9 [0,1]	1 [0,1]	71 [44,66]	61 [78,84]	0 [0,0]	0 [0,0]	0 [0,0]	1 [0,0]	8 [5,12]	14 [5,17]	0 [0,0]	2 [77,85]	0 [0,0]	67 [94,98]	65 [85,92]	98 [98,100]	90 [91,96]	28 [22,52]	34 [21,50]	96 [88,96]
task3	0 [0,1]	1 [0,1]	57 [29,70]	81 [42,68]	0 [0,0]	0 [0,0]	0 [0,0]	0 [0,2]	9 [5,10]	11 [7,11]	0 [0,0]	81 [13,51]	0 [0,0]	96 [71,98]	88 [85,94]	99 [99,100]	93 [89,98]	36 [6,28]	36 [6,11]	92 [86,93]
task4	0 [0,0]	0 [0,0]	50 [0,0]	56 [0,0]	0 [0,0]	0 [0,0]	0 [0,0]	1 [0,0]	7 [0,0]	9 [0,0]	0 [0,0]	31 [0,0]	0 [0,0]	88 [0,1]	89 [0,1]	99 [0,0]	94 [0,3]	15 [0,2]	4 [17,34]	89 [0,0]
task5	0 [9,20]	0 [1,9]	0 [44,60]	0 [54,65]	0 [0,0]	0 [20,20]	0 [20,20]	0 [20,20]	0 [24,26]	0 [25,28]	0 [0,0]	0 [34,45]	0 [0,0]	0 [64,76]	0 [64,73]	0 [79,79]	1 [73,78]	0 [17,34]	26 [33,43]	0 [74,77]
agg.	14	5	52	60	0	20	20	20	25	27	0	40	0	70	68	79	76	25	38	75
c4																				
task1	100 [100,100]	98 [96,100]	100 [78,90]	100 [100,100]	0 [0,0]	72 [0,1]	81 [37,66]	96 [53,77]	100 [52,70]	100 [29,70]	100 [96,98]	100 [98,100]	100 [80,91]	100 [64,90]	100 [98,100]	100 [99,100]	100 [95,99]	100 [99,100]	100 [100,100]	100 [100,100]
task2	100 [99,100]	59 [25,70]	85 [77,92]	100 [98,100]	0 [0,0]	0 [0,0]	52 [32,52]	66 [36,58]	62 [72,83]	50 [88,95]	97 [93,96]	97 [98,100]	99 [58,70]	85 [90,96]	80 [96,99]	99 [99,100]	100 [95,99]	97 [99,100]	100 [99,100]	100 [100,100]
task3	100 [94,100]	48 [31,72]	85 [24,75]	99 [99,100]	0 [0,0]	0 [0,0]	43 [0,0]	48 [0,1]	78 [49,65]	92 [89,96]	95 [96,98]	99 [100,100]	64 [46,65]	93 [72,90]	97 [99,100]	97 [98,100]	93 [89,96]	100 [98,100]	99 [89,96]	100 [100,100]
task4	98 [0,0]	51 [0,0]	50 [0,0]	99 [0,0]	0 [0,0]	0 [0,0]	0 [0,0]	0 [0,0]	57 [0,0]	93 [0,0]	97 [0,2]	100 [0,0]	54 [0,0]	82 [0,0]	99 [0,0]	99 [0,0]	92 [0,0]	100 [0,0]	100 [0,0]	100 [0,0]
task5	0 [79,80]	0 [44,59]	0 [59,69]	0 [79,80]	0 [0,0]	0 [10,17]	0 [30,40]	0 [39,45]	0 [56,62]	0 [62,71]	1 [77,78]	0 [79,80]	0 [58,63]	0 [67,74]	0 [79,80]	0 [79,80]	0 [76,77]	0 [80,80]	0 [79,80]	0 [80,80]
agg.	80	51	64	80	0	14	35	42	59	67	78	80	61	71	79	79	76	80	80	80

Table 8: **Per-task performance after 500K online environment steps (success rate, %).** Data sources, seed counts, confidence intervals, and highlighting follow Table 7. Best results in each row are highlighted, including ties.

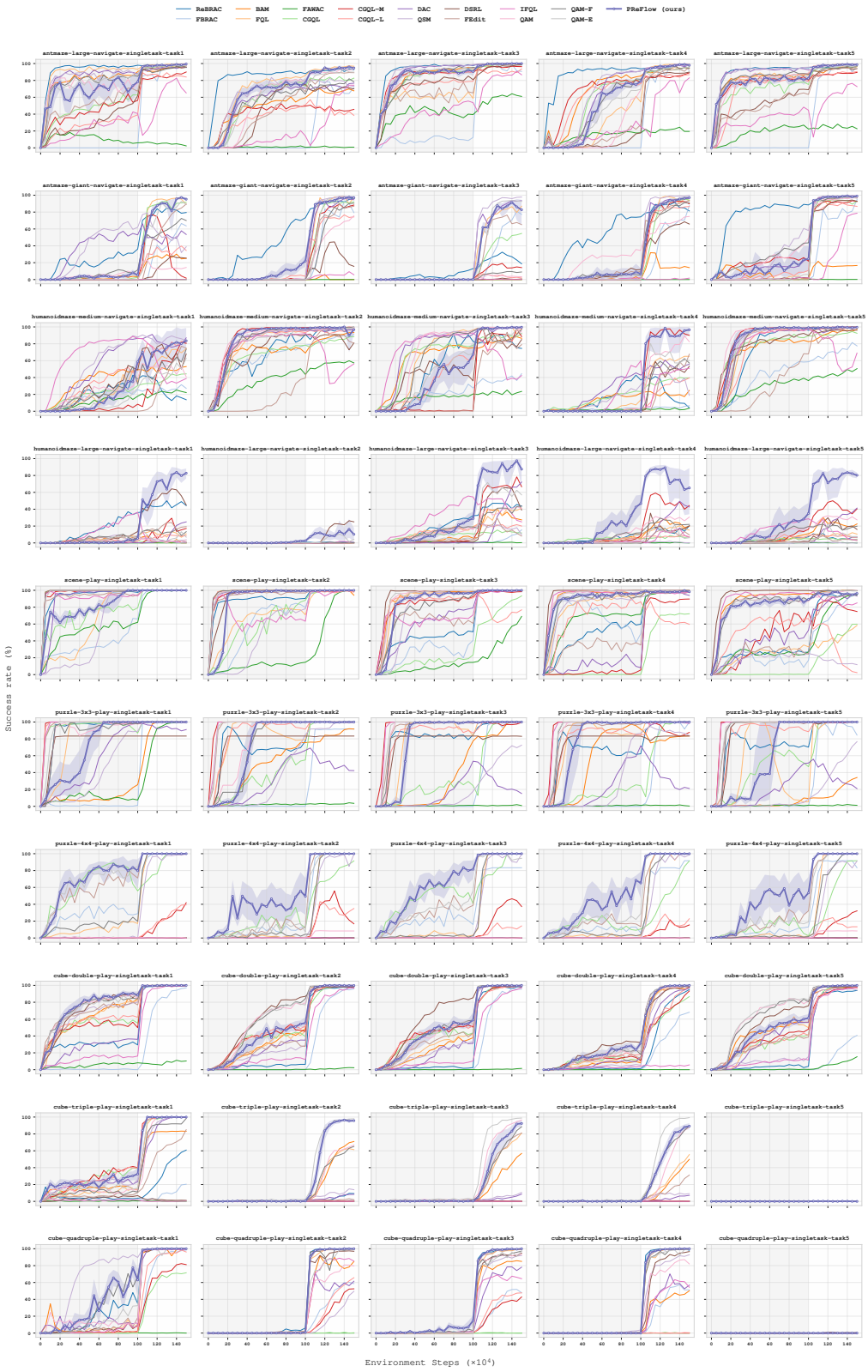


Figure 10: **Full training curves on the 50 OGBench tasks.** Online fine-tuning begins after 1M offline gradient steps and continues for 500K environment steps.