QPILOTS: EFFICIENT TEST-TIME Q-STEERING FOR FLOW POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching and diffusion policies are expressive action generators, but optimizing them with reinforcement learning (RL) is difficult: backpropagating the critic's action gradient through a multi-step denoising process is unstable, so existing methods distill the policy into a one-step actor or repeatedly fine-tune it as the critic improves. We show that inference-time steering alone, without ever training the policy against the critic, can outperform these fine-tuning approaches. Our method, QPILOTS, steers each denoising step of a flow policy: it maps the noisy intermediate action to an estimate of the clean action, where the critic is reliable, and adds the critic gradient computed there, rescaled to the magnitude of the base velocity. On a standard offline-to-online RL benchmark, QPILOTS achieves the best aggregate performance across 50 tasks, above baselines that fine-tune the policy against the critic. We also steer a frozen, pretrained VLA model, outperforming prior inference-time approaches in aggregate over six simulated manipulation tasks and on one real-robot task. We ablate several key design choices against practicality and performance.

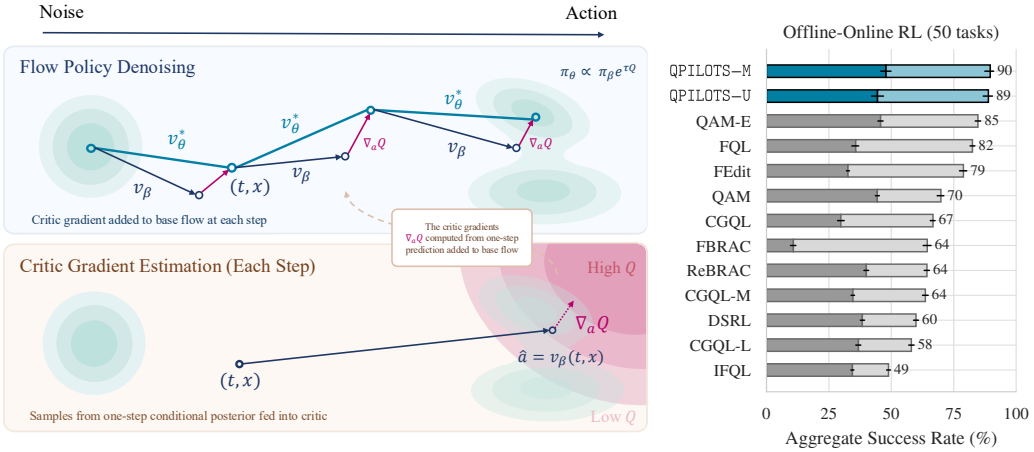


Figure 1: **QPILOTS steers a flow policy toward high-value actions at sampling time. Left:** Improving a base policy π_β with a critic Q while staying close to π_β yields the target $\pi^\star(a|s) \propto \pi_\beta(a|s)\exp(\tau Q(s,a))$. At each denoising step, QPILOTS adds to the base velocity v_θ the critic gradient evaluated at a one-step prediction of the final action, rescaled to the magnitude of v_θ, without updating the base policy. **Right:** Aggregated offline-to-online success rates on 50 OGBench tasks (Park et al., 2025a).

1 INTRODUCTION

Flow-matching and diffusion policies have become the de facto standard way to represent complex, multi-modal action distributions in continuous control and vision-language-action models (VLAs) (Lipman et al., 2023; Black et al., 2024; Chi et al., 2025), and there is sustained interest in

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.

improving these policies with downstream rewards (Park et al., 2025b; Wagenmaker et al., 2025; Dong et al., 2025; Li & Levine, 2026). This can be framed as *behavior-constrained* policy learning: a base flow policy $\pi_\beta(a|s)$ trained on offline data is improved with a learned critic Q without straying too far from π_β. Park et al. (2024) show that the main challenge is often *policy extraction*, obtaining an improved policy from (π_β, Q), rather than learning Q itself. An extractor may either modify the policy during training or steer its samples at inference time.

Improving a flow policy with a critic typically requires differentiating through its multi-step denoising process or changing its training objective (Wang et al., 2023; Park et al., 2025b; Li & Levine, 2026). Both can make adaptation costly, especially for large pretrained policies (Wang et al., 2026a). This motivates steering at *inference* time instead. However, on offline-to-online RL benchmarks, such methods have so far been outperformed by fine-tuning (Li & Levine, 2026): they act only through an auxiliary policy (Wagenmaker et al., 2025; Dong et al., 2025), are costly (Hansen-Estruch et al., 2023), or evaluate the critic at intermediate noisy actions, where it is unreliable, with a strength that varies across denoising steps and critic gradient scales (Li & Levine, 2026).

We show that this gap is not fundamental. We introduce `QPILOTS` (Q-guided Posterior Inference for Learning Off-policy and Test-time Steering), which adds a critic-gradient term to every denoising step of a flow policy that is never trained against the critic. `QPILOTS` evaluates the critic at a one-step prediction of the final action rather than at noisy actions it was never trained on, following training-free guidance (Chung et al., 2023; Bansal et al., 2024) and UF-OPS (Attarian et al., 2026). As our critic learns online, its gradient scale drifts: `QPILOTS` rescales the gradient to the magnitude of the base velocity, so that one steering coefficient stays meaningful across denoising steps and reward scales, and as the critic changes. Because the base flow and the critic are trained with their own standard objectives and combined only at sampling time, the same recipe applies to a policy trained from scratch and to a frozen, billion-parameter VLA, with a critic that keeps learning online from the steered policy's rollouts. We conduct a design study isolating which of these choices make steering outperform prior critic-guidance methods.

Contributions.

- **Steering can beat fine-tuning online:** competitive offline and above QAM-E and FQL at 89% after online training on 50 OGBench tasks (Park et al., 2025a; Li & Levine, 2026) (Section 5.1).
- **Scales to frozen VLAs:** $\pi_{0.5}$ (Black et al., 2025) rises from 26% to 77.5% with a critic learned online on LIBERO (DSRL: 59.8%) and to $38/50$ on a real robot (DSRL: $28/50$) (Sections 5.3 and 5.4).
- **What makes steering work:** a one-step clean-action estimate suffices, and rescaling makes steering easy to tune and robust to critic scale (Section 5.2).

2 Related Work

Offline-to-Online Reinforcement Learning. Offline-to-online RL methods leverage offline datasets to pre-train policies and value functions, then fine-tune them through online interaction to improve sample efficiency (Xie et al., 2021; Lee et al., 2021; Song et al., 2023b; Nakamoto et al., 2023; Tarasov et al., 2023; Zhou et al., 2025; Li et al., 2025a). A common recipe keeps the offline dataset in the replay buffer alongside new transitions (Ball et al., 2023), and offline pre-training tends to help on sparse-reward tasks (Li et al., 2025b; Li & Levine, 2026). We adopt this setup for the critic and focus on *policy extraction*, which selects high-value actions while staying close to the behavior support (Fujimoto & Gu, 2021; Kostrikov et al., 2022; Nakamoto et al., 2023) and can be the limiting factor even when the value function is well trained (Park et al., 2024).

RL with Diffusion and Flow Policies. The central challenge in RL with diffusion and flow policies is optimizing them against a critic through a multi-step denoising chain. We group existing approaches by how they treat the base policy.

1. Back-Propagating Through Time (BPTT) offers a straightforward way to use value signals by differentiating the critic through the full denoising chain to maximize Q (Wang et al., 2023; He et al., 2023; Zhang et al., 2024). Policy-gradient variants make likelihoods tractable for

flows (McAllister et al., 2025; Zhang et al., 2025). Gradients through long unrolled chains are prone to instability (Park et al., 2025b).

2. Policy Fine-Tuning without BPTT avoids full-chain gradients by changing the policy objective. Some methods distill the multi-step flow into a simplified one-step policy, improving stability but giving up part of the original model's iterative expressivity (Ding & Jin, 2024; Park et al., 2025b; Espinosa-Dice et al., 2025; Wang et al., 2026b). Other methods construct a step-wise training objective that uses critic information at intermediate denoising states. Methods like QAM (Li & Levine, 2026), alongside other gradient-approximated or value-weighted variants (Psenka et al., 2024; Fang et al., 2025; Frans et al., 2025), construct specialized losses where the critic gradient only passes through the base policy at isolated denoising steps. While these avoid the instability of backprop, they couple the base flow and the critic through a specialized loss and require re-training the policy whenever the critic changes.

3. Inference-time steering modifies the action-generation process at test time without impacting training. DSRL (Singh et al., 2021; Wagenmaker et al., 2025) trains a noise-space RL policy that perturbs the input latents of a frozen flow. Residual and edit policies (Johannink et al., 2019; Ankile et al., 2025; Dong et al., 2025; Yuan et al., 2025; Xiao et al., 2026) train a separate corrector that acts on completed actions. Best-of-N (Hansen-Estruch et al., 2023; Nakamoto et al., 2024; Mark et al., 2024; Attarian et al., 2026) samples N candidates from the base policy and picks the highest-Q one, scaling poorly with action dimension. Except for UF-OPS (discussed below), these consult the critic only on completed actions or through an auxiliary policy. `QPILOTS` instead injects a clean-action Q-gradient at *every* denoising step, without an auxiliary policy.

Flow Steering and Guided Generation. Classifier guidance (Dhariwal & Nichol, 2021) adds reward gradients to the diffusion drift, but evaluating the reward at noisy states biases the result because rewards are calibrated only on clean samples. Training-free guidance (Chung et al., 2023; Yu et al., 2023; Bansal et al., 2024) instead evaluates the reward at a one-step (Tweedie) estimate of the clean sample, and TFG (Ye et al., 2024) unifies these methods in one design space. Removing its remaining bias requires samples of the clean sample given the noisy one, obtained through particles (Wu et al., 2023; Du & Song, 2025), Monte Carlo samples around the estimate (Song et al., 2023a), or amortized posterior samplers such as Meta Flow Maps (Potaptchik et al., 2026; Holderrieth et al., 2026). We test both approaches with an RL critic as the reward and find that, for actions, the cheaper single estimate suffices. Closest to our work, UF-OPS (Attarian et al., 2026) already steers frozen diffusion policies with this gradient, using verifiers trained once on base-policy rollouts; we instead learn a TD critic online from the steered policy's own rollouts. Concurrently, QGF (Zhou et al., 2026) evaluates a similar algorithm in a purely offline setting.

3 Preliminaries and Background

Reinforcement Learning. We consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, R, \mu)$ with continuous actions $\mathcal{A} = \mathbb{R}^A$, transition kernel P, discount $\gamma \in [0, 1)$, reward R, and initial state distribution μ. We assume that we have access to an offline dataset $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^{|\mathcal{D}|}$ of transitions collected by a behavior policy π_β. The goal of offline RL is to learn a policy $\pi_\theta : \mathcal{S} \to \Delta\mathcal{A}$ from $\mathcal{D}$ that maximizes the expected discounted return $\mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k)\big]$ with $s_0 \sim \mu$, $a_k \sim \pi(\cdot|s_k)$, and $s_{k+1} \sim P(\cdot|s_k, a_k)$.

Flow Matching. A flow-matching generative model (Lipman et al., 2023; Liu et al., 2023b) learns a time-dependent velocity field $v_\theta : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ that transports a, typically Gaussian, noise distribution $p_0 = \mathcal{N}(0, I)$ to a data distribution p_1 via an ordinary differential equation (ODE)

$$\dot{x}_t = v_\theta(x_t, t), \qquad x_0 \sim p_0. \tag{1}$$

Under the linear interpolant $x_t = (1 - t)\, x_0 + t\, x_1$, the velocity field is trained with

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, x_0\sim p_0,\, x_1\sim p_1}\left[\|v_\theta(x_t, t) - (x_1 - x_0)\|^2\right], \tag{2}$$

known as the *conditional flow matching objective*. At inference, samples from p_1 are generated by sampling noise from the base distribution $x_0 \sim p_0$ and integrating the ODE via K Euler steps $x_{i+1} = x_i + h \cdot v_\theta(x_i, t_i)$, where $h = 1/K$, $t_i = ih$. To represent a state-conditioned policy $\pi_\theta(\cdot \mid s)$ in the RL setting, we condition the velocity field on the current state s, writing $v_\theta(x_t, t, s)$, and treat the integrated endpoint x_1 as the agent's action.

Stochastic Formulation. The deterministic ODE maps each noise x_0 to a unique endpoint. For any diffusion schedule $\sigma_t \geq 0$ and Brownian motion B_t, the SDE

$$dX_t = \left[v_\theta(X_t, t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right] dt + \sigma_t \, dB_t, \qquad X_0 \sim p_0, \tag{3}$$

has the same marginals p_t as the ODE (Song et al., 2021b), and induces a conditional distribution $p_{1|t}(x_1 \mid x_t)$ over endpoints reachable from x_t. We use the memoryless schedule $\sigma_t^2 = 2(1-t)/t$ for the linear interpolant (Domingo-Enrich et al., 2025), under which X_1 is independent of X_0 ($p_{1|0} = p_1$), so the initial noise needs no tilting.

Exponentially Tilted Generative Sampling. We next review a generic recipe for steering a generative model toward high-scoring samples. This view is common in reward-guided generation, classifier guidance, and control-as-inference (Song et al., 2021b; Potaptchik et al., 2026; Mihatsch & Neuneier, 2002). Let p_1 denote the terminal distribution produced by a base generative model, and $\psi : \mathbb{R}^d \to \mathbb{R}$ be a scalar utility assigned to a completed sample. *Exponentially tilted sampling* seeks to sample from $p_\psi(x) \propto p_1(x)\, e^{\psi(x)}$. In our setting, x_1 is an *action* and ψ is a critic-induced utility (Section 4). To realize this target with a stochastic process, define the *log-tilt potential*

$$V_t(x_t) := \log \mathbb{E}_{x_1 \sim p_{1|t}(x_1|x_t)} \left[e^{\psi(x_1)} \right], \tag{4}$$

which measures how much high-score terminal probability is reachable from the intermediate state x_t. (Despite the symbol, V_t is *not* the RL value function of Section 3.) Adding a drift correction proportional to ∇V_t to the base SDE yields the *steered probability-flow ODE* (Potaptchik et al., 2026)

$$\dot{x}_t^\star = v_\theta(x_t^\star, t) + \tfrac{\sigma_t^2}{2} \nabla V_t(x_t^\star), \qquad x_0^\star \sim p_0, \tag{5}$$

whose endpoint distribution at $t = 1$ is exactly p_ψ. The challenge is estimating ∇V_t, since $p_{1|t}(x_1 \mid x_t)$ is generally not available in closed form.

4 QPILOTS: Q-Guided Posterior Inference for Learning Off-Policy and Test-Time Steering

We now present our approach for steering a base flow policy with a learned critic. Rather than coupling policy and critic through specialized training-time losses, QPILOTS treats the base flow π_β and critic as independent components, each obtained via their own standard objectives, and combines them only at action-generation time. The base flow can be trained from scratch or taken off the shelf as a frozen pretrained policy. Algorithm 1 summarizes the procedure: our default, QPILOTS-U, evaluates the critic at a clean-action estimate and rescales its gradient, and QPILOTS-M, which replaces the estimate with learned posterior samples, serves as a reference.

Inference-Time Q-Steering. Following standard relative-entropy policy search (Peters et al., 2010; Peng et al., 2019; Abdolmaleki et al., 2018; Park et al., 2025b; Li & Levine, 2026), we cast critic-guided policy extraction as a KL-regularized improvement step. The goal is to maximize $\mathbb{E}_{a \sim \pi(\cdot|s)}[Q(s,a)]$ subject to $D_{\mathrm{KL}}(\pi(\cdot|s) \,\|\, \pi_\beta(\cdot|s)) \leq \delta(s)$. The closed-form solution is the exponentially tilted policy $\pi^\star(a|s) \propto \pi_\beta(a|s) \exp(\tau(s) Q(s,a))$ for an appropriate Lagrange multiplier $\tau(s) > 0$. To curb overestimation we use a pessimistic ensemble critic

$$\bar{Q}(s,a) := \tfrac{1}{J} \textstyle\sum_{j=1}^{J} Q_{\phi_j}(s,a) - \rho \, \mathrm{std}_j\big(Q_{\phi_j}(s,a)\big), \tag{6}$$

with ensemble size J and pessimism weight ρ matching Li & Levine (2026). Substituting $\bar{Q}$ for Q gives the tilted target of Section 3 with utility $\psi_s(a) := \tau \bar{Q}(s,a)$, so with drift $v_\theta(\cdot, t, s)$ the steered ODE of Eq. (5) has marginal $\pi^\star(\cdot|s)$ at $t = 1$. The remaining task is estimating $\nabla_{x_t} V_t(x_t; s)$, the gradient of Eq. (4) with $\psi = \psi_s$ and posterior $p_{1|t}(\cdot \mid x_t, s)$.

Evaluating the Critic at a Clean-Action Estimate. Classifier guidance evaluates the critic at the noisy latent, $\nabla_{x_t} \bar{Q}(s, x_t)$, where an off-policy critic trained on clean actions is unreliable: on five OGBench tasks, the spread of the critic ensemble at x_t is 1.4–2.3$\times$ its spread at dataset actions,

Algorithm 1 QPILOTS: Inference-Time Q-Steering for Flow Policies

Require: Observation s, base velocity field v_θ, critic ensemble $\{Q_{\phi_j}\}_{j=1}^{J}$, log-tilt-potential gradient estimator GRADV, steering coefficient α, flow steps K

$h \leftarrow 1/K$; $x_0 \sim \mathcal{N}(0, I)$
for $i = 0, \dots, K-1$ **do**
  $t \leftarrow i/K$; $\hat{v} \leftarrow v_\theta(x_i, t, s)$ ▷ Base velocity
  if $i \geq 0$ **then**
    $\widehat{\nabla V}_t \leftarrow \textsc{GradV}(x_i, t, s)$ ▷ QPILOTS-U (default): Eq. (7); QPILOTS-M: Eq. (9)
    $g \leftarrow \frac{\|\hat{v}\|}{\|\widehat{\nabla V}_t\| + \varepsilon} \widehat{\nabla V}_t$; $\hat{v} \leftarrow \hat{v} + \alpha \cdot g$ ▷ Rescale gradient and compute steering direction
  end if
  $x_{i+1} \leftarrow x_i + h \cdot \hat{v}$ ▷ Steer intermediate action
end for
return $a = \mathrm{clip}(x_K, -1, 1)$ ▷ Final action

against 1.0–1.4× at the clean-action estimate below (Section D.3). Following training-free guidance (Chung et al., 2023) and UF-OPS (Attarian et al., 2026), our default estimator, QPILOTS-U, instead replaces the expectation in Eq. (4) with a point estimate at the posterior mean. For the linear interpolant, the Tweedie (MMSE) denoiser (Bansal et al., 2024) is available in closed form from the base velocity, $\hat{x}_1 := x_t + (1-t)\, v_\theta(x_t, t, s) \approx \mathbb{E}[X_1 \mid X_t = x_t, s]$, and approximating $V_t(x_t; s) \approx \tau \bar{Q}(s, \hat{x}_1)$ gives

$$\widehat{\nabla V}_t^{\mathrm{U}}(x_t) = \tau\, J_t^\top\, \nabla_a \bar{Q}(s, a)\big|_{a=\mathrm{clip}(\hat{x}_1, -1, 1)}, \qquad J_t := \frac{\partial \hat{x}_1}{\partial x_t} = I + (1-t)\, \frac{\partial v_\theta(x_t, t, s)}{\partial x_t}, \quad (7)$$

computed with a single extra forward-backward pass through v_θ and the critic, with the clip applied straight-through. QPILOTS-U adds no training beyond the base flow and the critic. Unlike UF-OPS's fixed verifier, our TD critic changes during training: its gradient scale relative to v_θ varies by a median of 3.6× within a run (Section D.3), which the rescaling below addresses.

Gradient Rescaling. The theoretical drift $\frac{\sigma_t^2}{2}\nabla V_t$ uses $\sigma_t^2 = 2(1-t)/t$, which diverges as $t \to 0$ and vanishes as $t \to 1$. Because the schedule is memoryless, $p_{1|0} = p_1$ does not depend on x_0, so $\nabla V_0 = 0$ while σ_0^2 diverges; as is common with endpoint-singular schedules (Song et al., 2021a), we skip steering at $t = 0$. For $t > 0$ we follow Potaptchik et al. (2026) and rescale the gradient to match the base-drift magnitude:

$$v_\theta^{\mathrm{steered}}(x_t, t, s) = v_\theta(x_t, t, s) + \alpha\, g_t, \qquad g_t = \frac{\|v_\theta(x_t, t, s)\|}{\|\widehat{\nabla V}_t(x_t)\| + \varepsilon}\, \widehat{\nabla V}_t(x_t), \quad (8)$$

with $\varepsilon > 0$ a small constant. The rescaled drift keeps the steering direction, decouples its magnitude from the scale of the critic, which drifts during TD learning, and makes a single steering coefficient α meaningful across flow times. It no longer targets $\pi^\star$ for a fixed τ (Section D.4): for QPILOTS-U, τ cancels in g_t, so α alone sets the steering strength.

When Is the Point Estimate Enough? The chain rule in Eq. (7) passes the critic gradient through the Jacobian J_t of the denoiser, which has a clear probabilistic meaning. For the linear interpolant, differentiating Tweedie's formula gives $J_t = \frac{t}{(1-t)^2}\, \Sigma_t$ (Meng et al., 2021; Boys et al., 2024), where $\Sigma_t := \mathrm{Cov}[X_1 \mid X_t = x_t, s]$. If the posterior is Gaussian and $\bar{Q}$ is locally linear over it, then $\nabla_{x_t} V_t = \frac{t}{(1-t)^2}\, \Sigma_t\, \tau \nabla_a \bar{Q}(s, \hat{x}_1) = \widehat{\nabla V}_t^{\mathrm{U}}$ exactly (Appendix B). The point estimate is therefore biased only through posterior non-Gaussianity and critic curvature. Dropping J_t, as in the concurrent QGF (Zhou et al., 2026), is cheaper; we test this choice in Section 5.2.

Does a Learned-Posterior Sampler Help? To measure what removing this bias buys, we build QPILOTS-M, which draws N differentiable samples $\hat{X}_{0,1}(\epsilon_n; t, x_t, s)$ approximately from $p_{1|t}(\cdot \mid x_t, s)$ with a state-conditioned Meta Flow Map (MFM) (Potaptchik et al., 2026) and differentiates a Monte Carlo estimate of V_t:

$$\widehat{\nabla V}_t^{\mathrm{M}}(x_t) = \nabla_{x_t} \log \frac{1}{N} \sum_{n=1}^{N} \exp\Big(\tau\, \bar{Q}\Big(s,\, \hat{X}_{0,1}(\epsilon_n;\, t,\, x_t,\, s)\Big)\Big), \qquad \epsilon_n \overset{\mathrm{iid}}{\sim} \mathcal{N}(0, I). \quad (9)$$

Table 1: **Offline → Online Success Rate.** Each cell shows end-of-offline (gray) and end-of-online (colored) success rates with seed-level 95% bootstrap CIs in brackets. Bold marks the best offline and best online result per column.

	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] [70,79] **57** → 75	[65,74] [52,64] 69 → 58	[15,20] [19,27] 17 → 23	[57,73] [98,99] 65 → 99	[72,85] [100,100] 79 → **100**	[0,0] [0,0] 0 → 0	[8,10] [96,98] 9 → 97	[0,1] [9,19] 1 → 14	[6,11] [79,80] 9 → **80**	[39,41] [63,65] 40 → 64
BACKPROP											
FBRAC	[1,4] [82,95] 2 → 89	[0,0] [71,88] 0 → 81	[37,41] [45,53] 39 → 49	[0,0] [3,4] 0 → 4	[41,60] [97,100] 50 → 99	[0,1] [89,100] 0 → 95	[11,20] [85,98] 15 → 92	[0,0] [76,83] 0 → 79	[0,1] [1,9] 0 → 5	[0,0] [46,57] 0 → 51	[10,12] [63,66] 11 → 64
FINE-TUNING											
FQL	[72,79] [99,99] 76 → **99**	[0,0] [87,97] 0 → 93	[63,73] [83,91] 68 → 87	[7,11] [14,17] 9 → 16	[72,85] [87,96] 79 → 91	[59,80] [100,100] 70 → **100**	[3,8] [100,100] 5 → **100**	[43,49] [100,100] 46 → **100**	[2,4] [54,64] 3 → 60	[1,4] [79,80] 2 → **80**	[34,37] [82,83] 36 → 82
QAM	[78,84] [97,98] 81 → 98	[14,22] [52,55] 18 → 53	[64,69] [78,87] 67 → 83	[9,14] [17,24] 11 → 21	[95,98] [100,100] 97 → **100**	[99,100] [100,100] 99 → **100**	[0,0] [0,8] 0 → 3	[62,66] [100,100] 64 → **100**	[3,4] [63,76] 3 → 70	[2,4] [67,74] 3 → 71	[44,45] [68,71] 44 → 70
QAM-E	[80,86] [95,97] 83 → 96	[0,3] [91,97] 1 → **95**	[54,63] [75,90] 59 → 83	[1,3] [10,23] 2 → 16	[96,98] [100,100] 97 → **100**	[100,100] [100,100] **100** → **100**	[31,48] [97,100] **39** → 99	[63,68] [100,100] 65 → **100**	[4,6] [79,80] 5 → **79**	[4,9] [79,80] 6 → 79	[45,47] [84,86] 46 → 85
STEERING											
DSRL	[56,66] [88,91] 61 → 90	[1,4] [38,43] 3 → 40	[48,57] [70,82] 53 → 77	[2,5] [20,39] 3 → 29	[99,100] [100,100] **99** → **100**	[78,95] [77,95] 87 → 87	[0,0] [0,0] 0 → 0	[72,76] [98,99] 74 → 99	[1,2] [0,0] 1 → 0	[2,3] [77,78] 2 → 78	[38,39] [59,61] 38 → 60
IFQL	[32,39] [72,80] 36 → 76	[0,2] [16,18] 1 → 17	[85,87] [45,58] **86** → 51	[21,27] [6,9] **24** → 7	[80,88] [95,98] 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] [59,63] 2 → 61	[34,35] [48,50] 34 → 49
FEdit	[53,62] [95,96] 58 → 96	[1,3] [83,90] 2 → 86	[20,23] [69,90] 22 → 80	[2,3] [5,11] 3 → 8	[54,70] [98,99] 62 → 99	[98,100] [100,100] 99 → **100**	[26,41] [100,100] 34 → **100**	[37,43] [100,100] 40 → **100**	[2,3] [33,46] 2 → 40	[3,7] [79,80] 5 → **80**	[32,33] [77,80] 33 → 79
CGQL	[73,80] [94,95] 76 → 94	[0,2] [82,88] 0 → 85	[57,62] [69,73] 60 → 71	[4,5] [4,6] 5 → 5	[28,49] [80,89] 38 → 85	[40,55] [100,100] 48 → **100**	[16,34] [88,100] 24 → 95	[36,42] [95,98] 38 → 97	[7,9] [20,20] **8** → 20	[0,0] [10,17] 0 → 14	[28,31] [66,68] 30 → 67
CGQL-M	[68,73] [81,84] 71 → 82	[1,8] [55,60] 4 → 57	[40,43] [86,97] 42 → 92	[3,8] [27,40] 6 → 34	[65,84] [89,95] 74 → 92	[100,100] [96,99] **100** → 97	[0,0] [19,39] 0 → 29	[39,43] [97,98] 41 → 98	[7,9] [20,20] **8** → 20	[0,1] [28,41] 1 → 35	[34,35] [62,65] 35 → 64
CGQL-L	[62,67] [77,80] 65 → 79	[1,6] [55,59] 3 → 57	[57,67] [81,85] 62 → 83	[5,8] [7,10] 6 → 8	[84,92] [58,76] 88 → 67	[86,93] [100,100] 90 → **100**	[0,0] [16,34] 0 → 25	[43,47] [99,100] 45 → 99	[7,9] [20,20] **8** → 20	[0,1] [38,46] 0 → 42	[36,38] [57,59] 37 → 58
OURS											
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**

With exact posterior samples and the unrescaled drift, this estimator inherits the guarantees of Potaptchik et al. (2026); our practical sampler does not satisfy these conditions. The price is an auxiliary network to train and N MFM and critic passes per Euler step, so we use QPILOTS-M as a reference rather than as the default. Under rescaling, τ enters only its sample weights, and we fix $\tau = 1$. MFM background and training are deferred to Appendix A.

Training Settings. *(i) From scratch (OGBench):* v_θ and the critic are trained jointly on offline, then offline+online data with flow matching (Eq. (2)) and pessimistic TD learning, drawing next actions from the steered policy (Li & Levine, 2026). *(ii) Frozen base policy (LIBERO, DROID):* a public pretrained VLA stays frozen and only a task-specific critic is learned online. For QPILOTS-M, the MFM is trained alongside the flow on OGBench or distilled from the frozen VLA (Appendix A). Full procedures are in Appendix C.

5 EXPERIMENTS

5.1 OGBENCH: OFFLINE AND OFFLINE-TO-ONLINE (ALL COMPONENTS TRAINED FROM SCRATCH)

Setup. We evaluate on the OGBench (Park et al., 2025a) single-task benchmark, which spans 10 continuous-control domains (5 navigation, 5 manipulation) with 5 tasks per domain, for a total of 50 tasks. Each method first trains for 10^6 offline gradient steps on the OGBench dataset, then performs 5×10^5 online steps that interleave environment interaction (one gradient update per one environment interaction) with gradient updates over the offline data and an online replay buffer. Manipulation domains use action chunking, navigation domains use single-step actions, per-domain (H, γ, ρ) follow Li & Levine (2026) (Appendix C.4). Critic pessimism is set to $\rho = 0.5$ throughout,

except on the humanoid-maze domains, where we follow Li & Levine (2026) and use $\rho = 0$. We report mean evaluation success rate at the end of training over 8 seeds and 50 evaluation episodes per seed, with 95% bootstrap CIs across seeds shown in brackets.

Baselines. We group baselines by how they extract a policy from (π_β, Q): (i) the Gaussian-policy baseline ReBRAC (Tarasov et al., 2023); (ii) the backprop-based flow extractor FBRAC (Wang et al., 2023), which differentiates through the denoising chain; (iii) fine-tuning extractors that update the base flow with a critic-aware objective, including the one-step distillation FQL (Park et al., 2025b) and the adjoint-matching methods QAM and QAM-EDIT (QAM-E) of Li & Levine (2026); (iv) inference-time methods that wrap a frozen flow: latent-space steering DSRL (Wagenmaker et al., 2025), Best-of-32 IFQL (Hansen-Estruch et al., 2023), the Gaussian edit policy FEdit (Dong et al., 2025; Yuan et al., 2025), and classifier guidance CGQL (Dhariwal & Nichol, 2021) with its MSE/LinEx variants CGQL-M/L (Li & Levine, 2026). Baseline results are the released runs of Li & Levine (2026) (12 seeds), which retrain the critic for every task, seed, and method with the same critic learning algorithm and hyperparameters that we use. For our method, we report the default `QPILOTS-U` (Tweedie estimate with gradient rescaling) and the learned-posterior reference `QPILOTS-M` used in the design study (Section 5.2). Both share the base flow, critic ensemble, and Adam optimizer of Li & Levine (2026). The steering coefficient α is tuned per domain with `QPILOTS-M` following Li & Levine (2026)'s protocol, using two tuning tasks per domain (tasks 1 and 4 for navigation, tasks 2 and 4 for manipulation), 4 seeds per configuration. The selected value is reused for `QPILOTS-U` and across all five tasks of that domain, and evaluated on 8 fresh seeds. The sweep grid and selected values are in Appendix C.4.

Results. Steering outperforms fine-tuning after online training. Offline, `QPILOTS-U` matches QAM and QAM-E (44% vs. 44–46%). Online training brings it to 89% without updating the policy against the critic, above every training-time extractor: QAM-E (85%), FQL (82%), QAM (70%), and FBRAC (64%) (Table 1). Freezing the base flow after offline training, as with a VLA, changes mean online success by only 2 points (Section D.5). Gains are largest on long-horizon `humanoidmaze` domains, where `QPILOTS` improves on the closest baseline by 25 absolute points on `humanoidmaze-large`. Among inference-time methods, `QPILOTS` outperforms DSRL (60%), FEdit (79%), and all classifier-guidance variants (CGQL, 58–67%). CGQL uses the same base-flow architecture and critic learning algorithm; Section 5.2 attributes the gap to how the guidance enters the drift. Learning curves for all 50 tasks are in Fig. 7.

5.2 Design Study: What Matters When Steering with a Critic?

We vary `QPILOTS` along three axes on OGBench (Table 2): (1) where the critic is evaluated, (2) how accurately the posterior over clean actions is estimated, and (3) how the gradient enters the drift. For axis 1, `QPILOTS-N` is `QPILOTS-U` with the critic evaluated at the noisy x_t, as in classifier guidance; for axis 2, we compare with `QPILOTS-M`. We use the two tuning tasks per domain from Section 5.1 and three lower-quality `noisy` datasets (Park et al., 2025a). All variants share the base flow, the critic training, and the per-domain α of Section 5.1, tuned with `QPILOTS-M`.

Axis 1: the clean-action estimate matters on lower-quality data. On play data, evaluating the critic at the Tweedie estimate instead of at x_t changes little (91% $\rightarrow$ 93% online), with the clearest gain on the hardest domain, `cube-triple` (57% $\rightarrow$ 83%). On noisy data, where x_t lies further from well-supported actions, it becomes important: it raises success from 49% to 58% offline and from 81% to 99.5% online (2–14% vs. 98–100% on every seed of `cube-double-noisy` task 4).

Axis 2: posterior accuracy buys little. The learned-posterior reference `QPILOTS-M` helps somewhat offline on play data (38% $\rightarrow$ 43%), but the online gain over all 50 tasks is one point (89% $\rightarrow$ 90%; Table 1). On noisy data `QPILOTS-M` is worse than `QPILOTS-U` both offline (53% vs. 58%) and online (91% vs. 99.5%), and on LIBERO (Section 5.3) it is worse (72.5% vs. 77.5%) at twice the latency (Table 7). The MFM samples only approximate $p_{1|t}$, so this conclusion is bounded by the accuracy of the learned MFM. Our diagnostics suggest that the critic-curvature contribution is small: under the base-flow posterior, $\mathbb{E}[\bar{Q}(s, X_1)]$ and $\bar{Q}(s, \hat{x}_1)$ differ by at most $\approx$0.1 posterior standard deviations of $\bar{Q}$ on all five tasks we measured, at every t (Section D.3). Critic curvature and

Table 2: **Design ablations on OGBench** (mean success, %). *Play*: the 20 tuning tasks of all ten domains. *Noisy*: `cube-double` and `puzzle-3x3` tasks 2 and 4 and `scene` task 4 on the `noisy` datasets, the domains where `QPILOTS-U` is off the floor (it reaches at most 25% on `cube-triple`, `cube-quadruple`, and `puzzle-4x4 noisy`). `QPILOTS-M` uses $N = 16$ on Noisy. Offline at 1M steps, online at 1.5M (`QPILOTS-U` on `puzzle-3x3` task 4: 0.6M, all seeds at 100%). CGQL from Li & Levine (2026) (12 seeds; not reported on noisy data). Ours: 4–8 seeds.

			Play (20 tasks)		Noisy (5 tasks)	
Method	Critic evaluated at	Guidance	Offline	Online	Offline	Online
CGQL	noisy x_t	CG velocity	22.4	60.7	–	–
CGQL-M	noisy x_t (noise-cond. critic)	CG velocity	25.6	57.8	–	–
CGQL-L	noisy x_t (noise-cond. critic)	CG velocity	30.5	55.4	–	–
`QPILOTS-N`	noisy x_t	rescaled	40.7	91.1	49.0	80.6
`QPILOTS-U`	Tweedie $\hat{x}_1$	rescaled	37.8	93.3	**58.3**	**99.5**
`QPILOTS-M`	MFM posterior samples	rescaled	**43.2**	**95.4**	52.8	91.3

posterior non-Gaussianity (Section 4) should matter less for actions (5–70 dimensions, few modes) than for images (10^4–10^5), where MFMs show clear gains (Potaptchik et al., 2026).

Axis 3: rescaling shrinks the tuning range and absorbs changes in critic gradient scale. Without rescaling, the steering term scales with $\|\nabla_a \bar{Q}\|$, so the equivalent coefficient varies 30× across the ten domains and a median of 3.6× within a run (Section D.3). The tuned rescaled α spans one order of magnitude (0.2–2.0); the tuned coefficients of QAM, FQL, and FBRAC span 30×, 300×, and over 3000× (Li & Levine, 2026). The unrescaled σ_t drift also needs a wider search: on five tasks we swept its coefficient over 0.1–12, and on `cube-triple` and `humanoidmaze-large` no value we tried came close to rescaled steering (11% and 16% vs. 77% and 76%; Section D.4), whereas with rescaling one six-value grid serves all ten domains and all three variants. With the same critic learning algorithm, `QPILOTS-N` beats CGQL (Li & Levine, 2026) by 18 points offline and 30 online (95% CIs $[15.8, 20.9]$, $[27.9, 32.8]$) and CGQL-M/L on 16 of 20 tasks. Since both evaluate the critic at x_t, we attribute the gap to how the guidance enters the drift, although the CGQL runs come from the codebase of Li & Levine (2026) and our runs do not separate rescaling from CGQL's other differences (Section D.4).

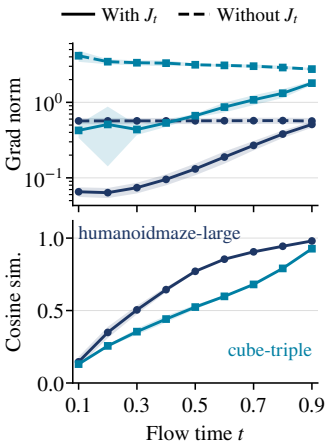


Figure 2: **Jacobian effect.** Unrescaled gradient norm with and without J_t (top) and their cosine similarity (bottom); mean ± s.d. over 4 seeds (Section D.6).

Additionally, rescaling absorbs the variation in gradient magnitude introduced by the denoiser Jacobian across flow steps. Figure 2 (top) shows that the gradient norm without J_t is relatively flat, whereas with J_t it grows toward the end of the flow; dropping J_t enlarges the gradient by ≈10–14× at the first steps but only ≈1–2× at the last. Rescaling removes this magnitude difference, leaving the direction difference shown at the bottom. With the step size fixed by rescaling, keeping or dropping the Jacobian ties on six of eight tasks (Section D.6).

5.3 LIBERO: Online RL (Frozen Base Policy, Learned Critic Ensemble)

Setup. We test whether `QPILOTS` can steer a large frozen pretrained policy, where full fine-tuning is expensive (Chen et al., 2025; Lu et al., 2025). The frozen base flow is the official $\pi_{0.5}$-LIBERO checkpoint (Black et al., 2025), pretrained on a closed-source robotics mixture and fine-tuned on LIBERO-40. We evaluate on six tasks from the larger LIBERO-90 suite (Liu et al., 2023a) on which $\pi_{0.5}$'s zero-shot success rate is between 5 and 50 percent. Per-task descriptions are in Appendix C.4. For `QPILOTS-M`, we distill the auxiliary meta flow map on the public LIBERO-90 demonstrations (Appendix A), keeping the $\pi_{0.5}$-LIBERO weights frozen. Each method runs 5×10^5 online environment steps with action chunking matching the base policy. We report results over

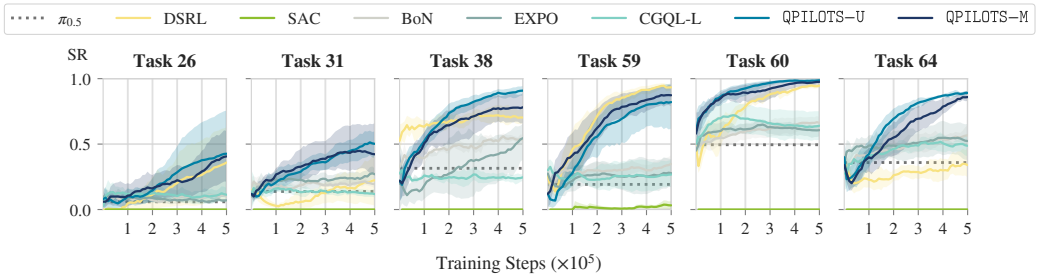


Figure 3: **LIBERO Online RL.** Per-task success rate over training for the LIBERO benchmark. Curves are time-weighted EMA-smoothed means over 5 seeds; shaded bands show seed-level 95% bootstrap CIs. Per-task descriptions, hyperparameters, and end-of-training numbers are in Section C.4.

5 seeds with 95% seed-level bootstrap confidence intervals. We follow the simulation setup of DSRL (Wagenmaker et al., 2025) but use SARSA targets, since sampling $a' \sim \pi^\star(\cdot|s')$ from a 3B-parameter model inside every critic update is prohibitive; SARSA beat the other feasible critic objectives we tried (Section D.2).

Baselines and Hyperparameters. We compare QPILOTS against DSRL (Wagenmaker et al., 2025) , Best-of-N (BoN, $N = 5$) action selection under the learned critic, EXPO (Dong et al., 2025), and CGQL-LinEx (Li & Levine, 2026). All baselines other than DSRL share the same SARSA critic for fairness. We tune one knob per method per task. For DSRL we sweep the action magnitude b_W over $\{1, 2, 3\}$, matching Wagenmaker et al. (2025). For QPILOTS and CGQL-LinEx we tune the steering coefficient α over $\{0.1, 0.2, 0.3\}$. Selected per-task values are listed in Table 5.

Results. Figure 3 shows per-task online learning curves, and Table 4 reports end-of-training success rates. QPILOTS-U raises the frozen $\pi_{0.5}$ from 26% to 77.5% average success, well above DSRL (59.8%), the strongest baseline and a leading method for steering frozen VLAs online. It improves over the base policy on all six tasks and is the best method on five (tasks 26, 31, 38, 60, 64), with gaps over DSRL ranging from 3 points on task 60 (within the bootstrap CIs) to 67 points on task 64. BoN, EXPO, and CGQL-LinEx, which either lack a directional gradient signal or evaluate the critic on noisy actions, plateau well below. Steering also adds only 4% inference latency to $\pi_{0.5}$, compared with 44% for Best-of-20 (Table 7).

5.4 Real-World: Steering a VLA on DROID

We steer the public $\pi_{0.5}$-DROID checkpoint (Black et al., 2025), kept frozen, with QPILOTS-U on a DROID (Khazatsky et al., 2024) Franka setup for the task *"put the stuffed animal in the box"*, learning the critic online from the steered rollouts. Without steering, $\pi_{0.5}$-DROID succeeds on $8/20$ trials (40%). Over 50 online episodes, the success rate of the steered rollouts rises from $4/10$ in the first ten episodes to $9/10$ in the last ten ($38/50$ overall; Fig. 4), and successful episodes become shorter, from 20 to 12 transitions on average. DSRL, trained online for the same 50 episodes, succeeds on $28/50$ ($7/10$ in the last ten; Appendix C.5).

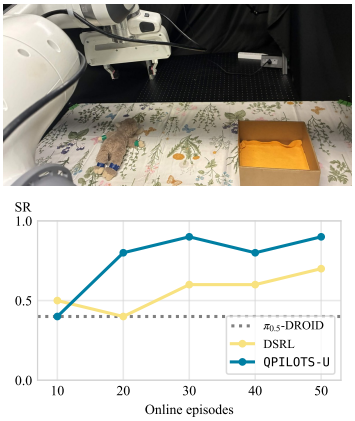


Figure 4: **Real robot.** Setup and success per 10 online episodes.

6 Discussion

QPILOTS steers a flow policy with a rescaled critic gradient at a clean-action estimate. It outperforms fine-tuning on OGBench after online training and steers a frozen 3B-parameter VLA, beating DSRL on LIBERO and on a real robot. Along the three axes of our design study, the Tweedie estimate is worth its negligible cost, especially on lower-quality data, a learned posterior sampler rarely pays off for actions, and rescaling reduces the per-domain/task coefficient search. **Limitations.** The steering coefficient α is tuned per domain, and our real-world evaluation is on a single task.

AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing methods, i.e., writing and debugging experiment code. We have not used generative AI tools to propose or refine research ideas or hypotheses, develop conceptual frameworks, design or give feedback on the research methodology or experiments, formulate mathematical claims or write proofs, or interpret results. Generating synthetic data, cleaning or reformatting datasets, qualitative data analysis, and translation are not applicable to this work. Additionally, we used generative AI tools to edit the paper for readability and to search for relevant literature and retrieve its bibliographic entries. We have reviewed all AI-assisted work: the authors reviewed and tested all AI-assisted code, checked all AI-suggested edits to the text, and verified every AI-suggested reference against its published source (arXiv or DBLP) before citing it. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies algorithms for improving pretrained flow policies with reinforcement learning. Simulation experiments use the public OGBench (Park et al., 2025a) and LIBERO (Liu et al., 2023a) benchmarks and the publicly released $\pi_{0.5}$-LIBERO and $\pi_{0.5}$-DROID checkpoints (Black et al., 2025). The real-world experiment uses a single robot arm in a laboratory setting under human supervision. No human subjects or personally identifiable data are involved. Because `QPILOTS` steers a frozen base policy toward actions preferred by a learned critic, errors in the critic can push the policy toward behavior that the base policy would not produce. Deploying such steering on physical robots therefore requires the same safety measures as other RL fine-tuning methods, such as constrained workspaces, human supervision, and conservative steering coefficients. We used such measures in our real-world experiment.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 specify the inference-time steering procedure and its design choices, Appendix A gives the Meta Flow Map background for `QPILOTS-M`, and Algorithm 2 gives the full offline-to-online training loop. Appendix C documents network architectures, optimizers, loss weights, inference settings, per-domain hyperparameters (Table 3), per-task LIBERO settings (Table 5), and compute requirements; Section D.7 reports the steering-coefficient sweep used for tuning. The Jacobian identity of Section 4 is derived in Appendix B, and the real-world setup is described in Appendix C.5. Anonymized code for the OGBench experiments is included in the supplementary material. This, and full code release for VLA experiments will also be available publicly before camera-ready.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2025.

Maria Attarian, Ian Vyse, Claas Voelcker, Jasper Gerigk, Evgenii Opryshko, Anas Almasri, Sumeet Singh, Yilun Du, and Igor Gilitschenski. Update-free on-policy steering via verifiers. *arXiv preprint arXiv:2603.10282*, 2026.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, pp. 1577–1594, 2023.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, pp. 17–40, 2025.

Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and Ömer Deniz Akyildiz. Tweedie moment projected diffusions for inverse problems. *Transactions on Machine Learning Research (TMLR)*, 2024.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2025. URL https://arxiv.org/abs/2510.25889.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *International Journal of Robotics Research*, 44:1684–1704, 2025.

Hyungjin Chung, Jeongsol Kim, Michael T. McCann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations (ICLR)*, 2023.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8780–8794, 2021.

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

Carles Domingo-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 *International Conference on Learning Representations (ICLR)*, 2025.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. *arXiv preprint arXiv:2507.07986*, 2025.

Max Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kianté Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 20132–20145, 2021.

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

Longxiang He, Li Shen, Linrui Zhang, Junbo Tan, and Xueqian Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *arXiv preprint arXiv:2310.05333*, 2023.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps. *arXiv preprint arXiv:2602.05993*, 2026.

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. IEEE, 2019.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

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

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2021.

Lu Li, Tianwei Ni, Yihao Sun, and Pierre-Luc Bacon. The three regimes of offline-to-online reinforcement learning. *arXiv preprint arXiv:2510.01460*, 2025a.

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

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023b.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL `https://arxiv.org/abs/2505.18719`.

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.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

Chenlin Meng, Yang Song, Wenzhe Li, and Stefano Ermon. Estimating high order gradients of the data distribution by denoising. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 25359–25369, 2021.

Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Machine learning*, 49 (2):267–290, 2002.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning (CoRL)*, pp. 4996–5013, 2024.

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

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

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

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

Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, pp. 1607–1612, 2010.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment. *arXiv preprint arXiv:2601.14430*, 2026.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via Q-score matching. In *International Conference on Machine Learning (ICML)*, pp. 41163–41182, 2024.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning (ICML)*, pp. 32483–32498, 2023a.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. In *The Eleventh International Conference on Learning Representations*, 2023b.

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

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 *Conference on Robot Learning*, pp. 258–282. PMLR, 2025.

Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, Xinlin Ren, Jingshun Huang, Mingjie Pan, Siyuan Feng, Zhi Chen, and Jianlan Luo. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. *arXiv preprint arXiv:2605.00416*, 2026a.

Zeyuan Wang, Da Li, Yulin Chen, Ye Shi, Liang Bai, Tianyuan Yu, and Yanwei Fu. One-step generative policies with Q-learning: A reformulation of MeanFlow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 26751–26759, 2026b.

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

Luhuan Wu, Brian L. Trippe, Christian A. Naesseth, David M. Blei, and John P. Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In *International Conference on Learning Representations (ICLR)*, 2026.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in Neural Information Processing Systems*, 34:27395–27407, 2021.

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. TFG: Unified training-free guidance for diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. FreeDoM: Training-free energy-guided conditional diffusion model. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23117–23127, 2023.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations (ICLR)*, 2025.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in neural information processing systems*, 37:98871–98897, 2024.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *The Thirteenth International Conference on Learning Representations*, 2025.

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 META FLOW MAP BACKGROUND FOR QPILOTS-M

Drawing exactly from $p_{1|t}(\cdot \mid x_t)$ would normally require an inner ODE solve starting from x_t, which is prohibitively expensive inside the outer Euler loop. A Meta Flow Map (MFM) (Potaptchik et al., 2026) amortizes this posterior sampling problem over contexts (t, x). For each context, it defines an auxiliary probability-flow ODE with drift $\bar{v}_u(\cdot, t, x)$ that transports exogenous noise $\epsilon \sim p_0$ to the endpoint posterior $p_{1|t}(\cdot \mid x)$. Here $u, w \in [0, 1]$ are auxiliary MFM times, distinct from the outer flow time t and the RL state s. The MFM learns the solution operators of this family with a single amortized network $\hat{X}_{u,w}(\bar{x}, t, x)$, parameterized in residual form

$$\hat{X}_{u,w}(\bar{x}; t, x) = \bar{x} + (w - u)\, \hat{v}_{u,w}(\bar{x}; t, x), \tag{10}$$

where $\hat{v}_{u,w}$ predicts the average velocity of the auxiliary trajectory between times u and w. Setting $u = 0$ and $w = 1$ gives the one-step posterior sampler

$$\hat{X}_{0,1}(\epsilon, t, x) = \epsilon + \hat{v}_{0,1}(\epsilon; t, x), \qquad \epsilon \sim p_0.$$

In the state-conditioned policy setting, we condition this sampler on the RL state and use $\hat{X}_{0,1}(\epsilon, t, x, s)$ as an approximate sample from $p_{1|t}(\cdot \mid x, s)$. Training combines two losses. The *diagonal loss* anchors the instantaneous velocity $\hat{v}_{u,u}$ to the conditional flow matching target:

$$\mathcal{L}_{\text{diag}} = \mathbb{E}_{t,u\sim\mathcal{U}[0,1]} \left[\left\| \hat{v}_{u,u}(\bar{I}_u; t, I_t) - (\bar{I}_1 - \bar{I}_0) \right\|^2 \right], \tag{11}$$

where $I_t = (1 - t)I_0 + tI_1$ and $\bar{I}_u = (1 - u)\bar{I}_0 + u\bar{I}_1$ are coupled interpolants sharing the data endpoint $I_1 = \bar{I}_1 \sim p_1$, with $I_0, \bar{I}_0 \sim p_0$ drawn independently. The *consistency loss* enforces the semigroup property $\hat{X}_{u,w} = \hat{X}_{m,w} \circ \hat{X}_{u,m}$ for $u < m < w$:

$$\mathcal{L}_{\text{cons}} = \mathbb{E}_{t,\, x,\, u<m<w,\, \bar{x}} \left[\left\| \hat{X}_{u,w}(\bar{x}; t, x) - \hat{X}_{m,w}\Big(\hat{X}_{u,m}(\bar{x}; t, x); t, x \Big) \right\|^2 \right]. \tag{12}$$

Efficient Gradient Estimation. With a trained MFM, the log-tilt-potential gradient can be estimated via the *MFM-G estimator* (Potaptchik et al., 2026):

$$\widehat{\nabla_{x_t} V_t}(x_t) = \nabla_{x_t} \log \frac{1}{N} \sum_{n=1}^{N} \exp\Big(\psi\Big(\hat{X}_{0,1}(\epsilon_n;\, t,\, x_t) \Big) \Big), \qquad \epsilon_n \sim p_0, \tag{13}$$

where $\hat{X}_{0,1}(\epsilon_n;\, t,\, x_t)$ is an MFM sample from the conditional endpoint distribution $p_{1|t}(x_1 \mid x_t)$. Differentiation through $\hat{X}_{0,1}$ with respect to x_t is tractable because $\hat{X}$ represents a single feedforward neural network evaluation. With an exact posterior sampler, as $N \to \infty$ the average converges to the posterior expectation in Eq. (4), so the MFM-G estimator converges to $\nabla_{x_t} V_t(x_t)$ under the assumptions of Potaptchik et al. (2026). Thus, MFM-G provides differentiable estimates of the steering direction without inner ODE integration.

In QPILOTS-M, we instantiate MFM-G with $\psi(\cdot) = \tau\bar{Q}(s, \cdot)$ and condition the MFM on the RL state, which yields Eq. (9). On OGBench, the MFM is trained jointly with the base flow and the critic using $\mathcal{L}_{\text{diag}}$ and $\mathcal{L}_{\text{cons}}$ (Section C.2). On LIBERO, the MFM is distilled from the frozen $\pi_{0.5}$ on LIBERO-90 demonstrations with the GLASS reparameterization of Potaptchik et al. (2026), using the frozen base flow as the conditional teacher. Unlike the original MFM setting, where the auxiliary network is trained on the same data that produced the base flow, only LIBERO-90 is available to us. It is a small subset of $\pi_{0.5}$'s non-public pretraining mixture, and the resulting distillation gap may contribute to QPILOTS-M trailing QPILOTS-U on LIBERO.

B TWEEDIE JACOBIAN AND POSTERIOR COVARIANCE

The Jacobian–covariance relation is a second-order Tweedie identity (Meng et al., 2021; Boys et al., 2024); we derive it for the linear interpolant for completeness. Under the linear interpolant, $X_t = tX_1 + (1-t)X_0$ with $X_0 \sim \mathcal{N}(0, I)$, so $p(x_t \mid x_1) = \mathcal{N}(tx_1, (1-t)^2 I)$ and $\nabla_{x_t} \log p_{1|t}(x_1 \mid x_t) = \frac{t}{(1-t)^2}\,(x_1 - \mathbb{E}[X_1 \mid x_t])$ (the x_1-independent normalizer contributes $-\nabla \log p_t(x_t)$, which cancels the mean). *Jacobian.* Differentiating $\hat{x}_1(x_t) = \mathbb{E}[X_1 \mid x_t] = \int x_1\, p_{1|t}(x_1 \mid x_t)\, dx_1$

gives $J_t = \partial \hat{x}_1 / \partial x_t = \mathbb{E}\Big[x_1 \nabla_{x_t} \log p_{1|t}^\top\Big] = \frac{t}{(1-t)^2}\,\Sigma_t$, where $\Sigma_t = \mathrm{Cov}[X_1 \mid x_t]$. For the exact velocity, $\hat{x}_1 = x_t + (1-t)v_t(x_t)$, so this is the Jacobian in Eq. (7). ***Log-tilt gradient.*** Similarly, $\nabla_{x_t} V_t = \mathbb{E}_{\tilde{p}}\big[\nabla_{x_t} \log p_{1|t}\big] = \frac{t}{(1-t)^2}\big(\mathbb{E}_{\tilde{p}}[X_1] - \hat{x}_1\big)$ with the tilted posterior $\tilde{p} \propto p_{1|t}\, e^{\psi}$. If $p_{1|t} = \mathcal{N}(\hat{x}_1, \Sigma_t)$ and $\psi(x_1) = \psi(\hat{x}_1) + g^\top (x_1 - \hat{x}_1)$, then $\tilde{p} = \mathcal{N}(\hat{x}_1 + \Sigma_t g, \Sigma_t)$ and hence $\nabla_{x_t} V_t = \frac{t}{(1-t)^2}\Sigma_t\, g = J_t^\top g$. This is exactly the full-Jacobian `QPILOTS-U` estimator. Deviations are caused by non-Gaussian (e.g., multimodal) posteriors and by curvature of ψ over the support of $p_{1|t}$, both of which are small when Σ_t is small.

C Implementation Details

The per-step computational overhead of inference-time steering (Algorithm 1) is a single forward-backward through v_θ and the critic for `QPILOTS-U`, and N forward passes through the MFM and critic plus one backward pass for `QPILOTS-M`–all single-step evaluations with no inner rollouts.

C.1 Architecture

In OGBench experiments, all networks are MLPs with hidden dimensions $(512, 512, 512, 512)$ and SiLU activations. The base velocity field $v_\theta(x_t, t, s)$ is an `ActorVectorField` that concatenates $[s, x_t, \mathrm{FF}(t)]$ and outputs a velocity in $\mathbb{R}^{H \cdot A}$, where $\mathrm{FF}(\cdot)$ is a 64-dim Fourier-feature embedding and H is the action-chunking horizon. The critic is a J=10-ensemble of `Value` networks (LayerNorm enabled) with pessimism weight ρ (Section 4). The Meta Flow Map $\hat{v}_{u,w}(\bar{x};\, t, x_t, s)$ is an `MFMFlowMap`: it concatenates $[s, \bar{x}, x_t, \mathrm{FF}(t), \mathrm{FF}(u), \mathrm{FF}(w)]$, with three independent Fourier embeddings for the outer time t and the auxiliary times (u, w). LayerNorm is disabled inside the MFM and base velocity field; its conditioning-action input is passed through unchanged (no t-gating). We instantiate Polyak-averaged target networks for the critic, the base flow, and the MFM.

In LIBERO experiments, the critic $Q(s, a)$ uses double-view $(64, 64)$ images from the global camera and the wrist camera as the pixel inputs. Its network consists of a 4-layer CNN encoder and MLPs with hidden dimensions $(128, 128, 128)$. We use the same 10-network Q-value ensemble as the critic in OGBench experiments. For baselines like DSRL or EXPO that require an additional Gaussian policy, we use the same structure as the critic and follow the hyperparameters used in OGBench.

C.2 Training

For OGBench, all parameters (base flow v_θ, MFM $\hat{v}_\xi$, and critic ensemble $\{Q_{\phi_j}\}$) are jointly optimized with a single Adam optimizer at learning rate 3×10^{-4} and global gradient clipping at norm 1.0. No learning-rate schedule or warmup is used. Targets are updated by Polyak averaging with $\tau_{\text{critic}} = \tau_{\text{flow}} = \tau_{\text{mfm}} = 0.005$. Each gradient step samples a length-H sequence batch of size 256 from the offline dataset (offline phase) or from the union of offline data and online replay (online phase, one gradient update per one environment interaction, no balanced sampling). Cross-episode transitions are masked out via the `valid` mask of Li & Levine (2026).

The total loss is

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{critic}}}_{\text{TD}} + \underbrace{\mathcal{L}_{\text{FM}}}_{\text{base flow, Eq. (2)}} + \lambda_{\text{diag}}\mathcal{L}_{\text{diag}} + \lambda_{\text{cons}}\mathcal{L}_{\text{cons}},$$

with $\lambda_{\text{diag}} = 1.0$, $\lambda_{\text{cons}} = 0.5$ for `QPILOTS-M` and $\lambda_{\text{diag}} = \lambda_{\text{cons}} = 0$ for `QPILOTS-U` (no MFM is trained). $\mathcal{L}_{\text{critic}}$ uses TD against the pessimistic ensemble target with the next action sampled from the steered policy (`critic_steered_targets`). $\mathcal{L}_{\text{diag}}$ samples the outer time as $t = u^2$, $u \sim \mathcal{U}[0, 1]$ to bias supervision toward small t where the posterior is broader (`mfm_t_outer_power`=2). $\mathcal{L}_{\text{cons}}$ samples ordered times $0 \le u < m < w \le 1$ by sorting two uniforms and taking $m = u + \gamma(w - u)$ with $\gamma \sim \mathcal{U}[0, 1]$, and supervises the direct map against the composed map computed under target parameters. Both MFM losses use the adaptive reweighting of Potaptchik et al. (2026) (`adaptive_loss`=true) with $p_{\text{diag}} = 0.5$, $p_{\text{cons}} = 1.0$, $c = 0.01$. We do not use the FT auxiliary loss in our experiments ($\lambda_{\text{ft}} = 0$).

LIBERO MFM Distillation. For LIBERO, the auxiliary meta flow map is pretrained offline against the frozen $\pi_{0.5}$-LIBERO checkpoint and the critic is then trained online (the base flow is

Algorithm 2 `QPILOTS-M`: Offline-to-Online Training (OGBench)

Require: Offline data $\mathcal{D}_{\text{off}}$, environment, offline steps M_{off}, online env steps M_{on}, batch size B, target rate τ, MFM weights $\lambda_{\text{diag}}, \lambda_{\text{cons}}$, steering coefficient α, action-chunk horizon H

Initialize base flow v_θ, critic ensemble $\{Q_{\phi_j}\}_{j=1}^{J}$, MFM $\hat{v}_\xi$ (`QPILOTS-M` only), and target copies (θ', ϕ', ξ'); replay $\mathcal{R} \leftarrow \emptyset$
// Offline phase
for $k = 1, \ldots, M_{\text{off}}$ **do**
  Sample length-H sequence batch $\mathcal{B} \sim \mathcal{D}_{\text{off}}$ of size B; call UPDATE($\mathcal{B}$)
end for
// Online phase (one gradient step per environment step)
Reset environment: $s \leftarrow s_0$
for $k = 1, \ldots, M_{\text{on}}$ **do**
  Sample chunk $a_{1:H} \sim \pi^\alpha(\cdot \mid s)$ via Algorithm 1 ▷ Steered rollout
  Execute $a_{1:H}$ open-loop; push transitions to $\mathcal{R}$; advance s (reset on terminal)
  Sample $\mathcal{B} \sim \mathcal{D}_{\text{off}} \cup \mathcal{R}$ of size B; call UPDATE($\mathcal{B}$)
end for
return v_θ, $\{Q_{\phi_j}\}_{j=1}^{J}$, $\hat{v}_\xi$

procedure UPDATE($\mathcal{B}$)
  For each $(s, a, r, s') \in \mathcal{B}$: sample $a' \sim \pi^\alpha(\cdot \mid s')$ via Algorithm 1 using target nets
  TD target: $y = r + \gamma\big(\frac{1}{J}\sum_j Q_{\phi'_j}(s', a') - \rho\,\mathrm{std}_j\, Q_{\phi'_j}(s', a')\big)$
  $\mathcal{L}_{\text{critic}} = \frac{1}{J}\sum_j \big(Q_{\phi_j}(s, a) - \mathrm{sg}(y)\big)^2$ ▷ Pessimistic ensemble TD
  $\mathcal{L}_{\text{FM}}$ from Eq. (2); $\mathcal{L}_{\text{diag}}, \mathcal{L}_{\text{cons}}$ from Eqs. (11) and (12)
  $\mathcal{L} = \mathcal{L}_{\text{critic}} + \mathcal{L}_{\text{FM}} + \lambda_{\text{diag}}\mathcal{L}_{\text{diag}} + \lambda_{\text{cons}}\mathcal{L}_{\text{cons}}$
  Joint Adam step on $(\theta, \{\phi_j\}, \xi)$ with global gradient clipping at norm 1.0
  Polyak: $\theta' \leftarrow (1-\tau)\theta' + \tau\theta$ (similarly for ϕ'_j and ξ')
end procedure

never updated). The MFM $\hat{v}_{u,w}$ uses a transformer-style architecture matching the $\pi_{0.5}$ action expert: hidden width 512, 8 blocks, and 8 attention heads (`mfm_width`=512, `mfm_num_blocks`=8, `mfm_num_heads`=8). Following Potaptchik et al. (2026), the MFM is initialized from the $\pi_{0.5}$-LIBERO action-expert weights with the auxiliary-time gating zero-initialized, so that at the start of distillation the network reproduces the frozen base flow exactly and only learns the off-diagonal posterior structure. Distillation follows the GLASS reparameterization of Potaptchik et al. (2026): the frozen $\pi_{0.5}$ acts as the conditional teacher, supplying noisy-state/clean-action pairs from LIBERO-90 demonstrations, and the MFM is trained with the same diagonal and consistency objectives ($\lambda_{\text{diag}} = 1.0$, $\lambda_{\text{cons}} = 0.5$, adaptive reweighting on) and the same outer-time power-2 sampling as in OGBench. We use batch size 256 for 4×10^4 gradient steps, optimized with RAdam at learning rate 1×10^{-4} with a linear warmup over 2000 steps and a constant schedule thereafter, and global gradient clipping at norm 1.5. The MFM target is updated by Polyak averaging with $\tau_{\text{mfm}} = 10^{-4}$ (matching Potaptchik et al. (2026)'s reference value, which is more conservative than the OGBench setting since the MFM is updated against a frozen teacher). The critic uses a separate Adam optimizer at 3×10^{-4} with a 500-step warmup and cosine decay over 3×10^5 steps to $0.1\times$ peak, with global gradient clipping at norm 1.0 and Polyak rate $\tau_{\text{critic}} = 0.005$.

Algorithm 2 summarizes the resulting offline-to-online procedure for the OGBench experiments. `QPILOTS-U` uses the same outer loop with $\lambda_{\text{diag}} = \lambda_{\text{cons}} = 0$ and no MFM network, in which case the steered policy in Algorithm 1 replaces $\widehat{\nabla V}_t^{\text{M}}$ (Eq. (9)) with the Tweedie estimator $\widehat{\nabla V}_t^{\text{U}}$ of Eq. (7).

C.3 INFERENCE

Action generation uses K=10 Euler steps and best-of-1 selection (`flow_steps`=10, `best_of_n`=1). For `QPILOTS-M` we draw N=8 posterior samples per Euler step (`num_mfm_samples`=8) in OGBench and N=4 posterior samples per Euler step (`num_mfm_samples`=4) in LIBERO. Both `QPILOTS-M` and `QPILOTS-U` use gradient

rescaling (Eq. (8)) rather than the $\sigma_t^2/2$ schedule, with steering skipped at $t = 0$. For `QPILOTS-M` the steering gradient is computed against the online MFM and the target critic; for `QPILOTS-U` the gradient is computed against the target base flow and the target critic, which we found to be more stable for the biased Tweedie estimate. At inference each H-action chunk is executed open-loop in the environment.

C.4 Per-Domain Settings

OGBench Per-Task Settings and Results. Domain-level hyperparameters follow Li & Levine (2026) and are summarized in Table 3. The steering coefficient α was selected per domain by sweeping $\{0.1, 0.2, 0.3, 0.5, 1.0, 2.0\}$ on two tuning tasks per domain (tasks 1 and 4 for navigation, tasks 2 and 4 for manipulation) with 4 seeds per configuration, mirroring the protocol of Li & Levine (2026). The combined performance across the two tuning tasks selects α for the domain, which is then reused across all five tasks and evaluated on 8 fresh seeds. Per-domain sweep curves are shown in Fig. 6. The selected coefficients are: `antmaze-large`: 0.2, `antmaze-giant`: 0.3, `humanoidmaze-medium`: 0.2, `humanoidmaze-large`: 0.2, `cube-double`: 0.3, `cube-triple`: 0.5, `cube-quadruple`: 0.5, `scene-play-sparse`: 0.2, `puzzle-3x3-play-sparse`: 0.3, `puzzle-4x4-play-sparse`: 2.0. Figure 7 reports per-task success rate across all 50 OGBench tasks for every method evaluated in Section 5.1.

Table 3: Per-domain hyperparameters: action-chunking horizon H, discount γ, critic pessimism ρ, and tuned steering coefficient α. All other hyperparameters (network sizes, optimizer, batch size, K, N, τ, λ_{diag}, λ_{cons}) are shared across domains.

Domain	H	γ	ρ	α
`antmaze-large-navigate`	1	0.99	0.5	0.2
`antmaze-giant-navigate`	1	0.995	0.5	0.3
`humanoidmaze-medium-navigate`	1	0.995	0.0	0.2
`humanoidmaze-large-navigate`	1	0.995	0.0	0.2
`cube-double-play`	5	0.99	0.5	0.3
`cube-triple-play`	5	0.99	0.5	0.5
`cube-quadruple-play`	5	0.99	0.5	0.5
`scene-play-sparse`	5	0.99	0.5	0.2
`puzzle-3x3-play-sparse`	5	0.99	0.5	0.3
`puzzle-4x4-play-sparse`	5	0.99	0.5	2.0

LIBERO Per-Task Settings and Results. Table 5 lists the six evaluation tasks, the natural-language goal description supplied to $\pi_{0.5}$, the zero-shot $\pi_{0.5}$-LIBERO success rate, and the per-task selected steering coefficient α (used by `QPILOTS-U`, `QPILOTS-M`, and CGQL-LinEx) and DSRL action magnitude b_W. Table 4 reports end-of-training success rate per task. The critic architecture is described in Section C.1; the LIBERO setting is pure online RL with no offline phase, BoN uses $N = 5$, and reported end-of-training success rates are raw (unsmoothed) means across seeds; only the learning curves in Fig. 3 are EMA-smoothed. The action-chunking horizon is $H = 10$, matching $\pi_{0.5}$; the discount is $\gamma = 0.99$ and critic pessimism is $\rho = 0.5$ across all 6 tasks.

Compute. OGBench experiments use TPU v4-8 hosts. `cube-quadruple-play` and `puzzle-4x4-play-sparse` additionally use the OGBench 100M datasets. A full 1.5×10^6-step run (offline + online) for one task and four seeds in parallel takes approximately 8 hours for `QPILOTS-U` and 12 hours for `QPILOTS-M`, depending on the task (different domains differ widely in evaluation time requirements). In online RL on LIBERO tasks, 5×10^5 takes about 6 hours for `QPILOTS-U` and 8 hours for `QPILOTS-M`. Offline MFM distillation on LIBERO (4×10^4 gradient steps, batch size 256) takes approximately two days on a single NVIDIA H100 due to the cost of running the frozen 3B-parameter $\pi_{0.5}$ teacher in every batch. This cost is paid once as we use the same MFM for all tasks.

Table 4: **LIBERO end-of-training success rate (%).** Mean over 5 seeds at 5×10^5 steps, without smoothing along the training axis, with seed-level 95% bootstrap CIs in brackets. Bold marks the best mean per column; the CIs of QPILOTS-U and DSRL overlap on tasks 26 and 60.

Method	Task 26	Task 31	Task 38	Task 59	Task 60	Task 64	Mean
$\pi_{0.5}$ (zero-shot)	6	14	32	19	50	36	26
DSRL	44 [15,72]	29 [0,58]	73 [60,90]	**94** [87,99]	96 [93,99]	22 [18,28]	59.8 [52.5,67.0]
BoN ($N = 5$)	9 [2,16]	16 [4,32]	52 [40,65]	41 [36,48]	67 [61,72]	61 [45,81]	41.2 [36.2,46.4]
EXPO	8 [2,12]	24 [17,31]	76 [59,89]	29 [19,38]	61 [57,66]	50 [30,70]	41.3 [35.6,46.7]
CGQL-LinEx	6 [0,12]	10 [2,20]	28 [19,34]	29 [18,42]	66 [52,80]	42 [31,61]	30.2 [25.6,35.0]
QPILOTS-U	**52** [22,82]	**51** [43,61]	**91** [86,96]	82 [58,98]	**99** [98,100]	**89** [81,96]	**77.5** [70.5,84.0]
QPILOTS-M	46 [21,72]	38 [19,67]	79 [58,95]	87 [75,99]	**99** [97,100]	86 [78,96]	72.5 [65.5,79.8]

Table 5: LIBERO-90 evaluation tasks. Zero-shot SR is the success rate of the frozen $\pi_{0.5}$-LIBERO checkpoint without any online learning. α is the steering coefficient used by QPILOTS-U, QPILOTS-M, and CGQL-LinEx; b_W is DSRL's action magnitude.

Task	Description	Zero-shot SR (%)	α	b_W
26	put the wine bottle in the bottom drawer of the cabinet	5.9	0.3	2
31	put the black bowl on top of the cabinet	13.8	0.3	3
38	put the right moka pot on the stove	31.5	0.1	1
59	pick up the tomato sauce and put it in the tray	19.2	0.3	2
60	pick up the black bowl on the left and put it in the tray	49.5	0.2	2
64	stack the right bowl on the left bowl and place them in the tray	35.9	0.2	1

C.5 Real-World DROID Setup

The base policy is the public $\pi_{0.5}$-DROID checkpoint (Black et al., 2025), which stays frozen throughout. The critic is trained online from the rollouts of the steered policy, and every online episode uses QPILOTS-U steering. As a baseline, DSRL (Wagenmaker et al., 2025) steers the same frozen checkpoint for 50 online episodes. Table 6 reports the success rates of both methods and the episode length of QPILOTS-U over the 50 online episodes. The unsteered base policy succeeds on 8 of 20 trials.

Protocol. Both methods follow the real-robot setup of the public DSRL implementation (Wagenmaker et al., 2025). Observations consist of one exterior and one wrist camera image, resized to 128×128, together with the joint and gripper positions and the frozen policy's VLM embedding. The policy is queried every 10 control steps, so each transition spans one action chunk and is discounted by γ^{10} with $\gamma = 0.99$. Each episode is capped at 40 action chunks. After each episode, the operator labels it as a success or a failure; a success means the stuffed animal ends inside the box. Following DSRL, the label is converted into a sparse reward of -1 per transition and 0 on the final transition of a successful episode. Both methods update between episodes with DSRL's schedule: 5,000 gradient steps after the first episode, then 30 gradient steps per collected transition, with batch size 256.

QPILOTS-U. The critic is the 10-network ensemble of the LIBERO experiments (Section C.1), trained with SARSA targets and pessimism $\rho = 0.5$. We fix the steering coefficient to $\alpha = 0.1$, the lowest value of the LIBERO grid, without tuning.

DSRL. DSRL trains a pixel-based SAC agent in the noise space of the frozen policy with the architecture and hyperparameters of its public real-robot configuration: action magnitude $b_W = 2.5$, two critics with min reduction, and 3×1024 MLPs on a small convolutional encoder.

Table 6: **DROID online steering** on *"put the stuffed animal in the box"*. Success counts in blocks of ten online episodes, and the mean length (transitions) of successful QPILOTS-U episodes. The unsteered $\pi_{0.5}$-DROID base policy succeeds on $8/20$ trials.

	Episodes	1–10	11–20	21–30	31–40	41–50	Total
DSRL	Successes	5/10	4/10	6/10	6/10	7/10	28/50
QPILOTS-U	Successes	4/10	8/10	9/10	8/10	9/10	38/50
QPILOTS-U	Mean successful length	20.3	13.0	14.3	14.6	12.4	14.3

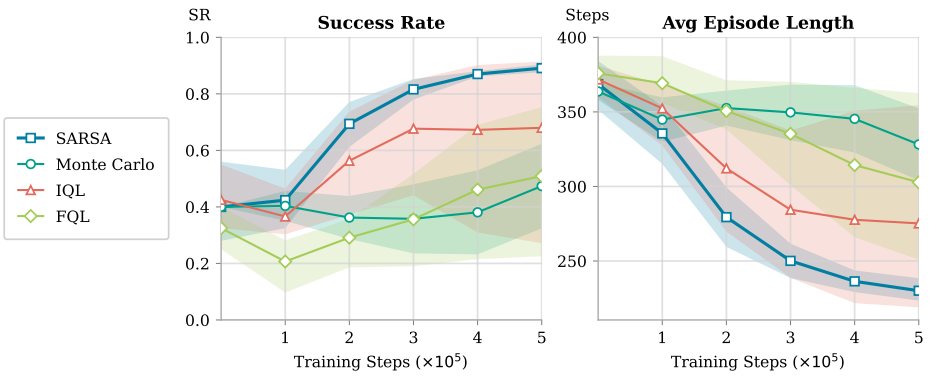


Figure 5: **Value Learning Method Ablation.** We compare SARSA with IQL, FQL, and Monte-Carlo on the LIBERO-90 task 64. It shows means over 5 seeds and shaded bands show seed-level 95% bootstrap CIs. A lower average episode length indicates that the method can complete the task faster.

D Additional Experiments

D.1 Latency Analysis

We conduct an inference-time latency analysis per time step using $\pi_{0.5}$ model. Given that $\pi_{0.5}$ is a 3B-parameter model, it inherently incurs a relatively high inference cost. When employing the Best-of-N, the repeated inference cycles lead to a surge in latency, making real-time control challenging. As shown in Table 7, we perform single-step inference on task 59 across methods running on a single NVIDIA RTX PRO 6000. Using $N = 20$ in Best-of-N results in 136.9% performance improvement with 1.44 times slower inference, but QPILOTS-U gets +333.8% with only 1.04 times overhead, even faster than Best-of-5.

Table 7: Comparison of success rate after training 500k update steps, inference time, and computational overhead. For the $\pi_{0.5}$ model using Best-of-N, the inference time escalates with N.

Method	N	Success rate (%)	Inference-time	Overhead
$\pi_{0.5}$	-	19.2	112.7ms	1.0×
Best-of-N	5	36.4 (+89.5%)	127.2ms	1.14×
	10	41.4 (+115.6%)	137.0ms	1.22×
	20	45.5 (+136.9%)	158.0ms	1.44×
QPILOTS-U	-	83.3 (+333.8%)	**114.1ms**	**1.04**×
QPILOTS-M	-	**88.5** (+360.9%)	240.1ms	2.14×

D.2 Value Learning Ablation

VLA's slow inference makes standard Bellman target-based TD-learning almost infeasible, as each update requires one policy query per sample in the batch. To find an alternative, we compare 4

value-learning objectives for the critic on LIBERO-90 task 64: SARSA (we choose for QPILOTS), Monte Carlo (MC) returns, Implicit Q-Learning (IQL) (Kostrikov et al., 2022), and Flow Q-Learning (FQL) (Park et al., 2025b). Note that FQL is not used for policy learning. Instead, we still use the steered $\pi_{0.5}$ for inference, but distill the steered $\pi_{0.5}$ policy into a one-step policy for Q-learning next action query.

As shown in Fig. 5, SARSA reaches a success rate of 90% at 5×10^5 training steps, outperforming the other 3 approaches. Despite being unbiased, the MC estimator regresses directly against discounted episode returns, which are dominated by large negative rewards in early training, leading to slower initial learning across seeds. IQL and FQL are originally designed for offline RL. However, under the non-stationary replay buffer of our online setting, both methods exhibit instability (high variance), suggesting that current offline value learning approaches still struggle with the continually shifting action distribution. In contrast, SARSA is shown as a pragmatic and effective choice for online critic training under a generalist policy.

D.3 Critic Reliability, Critic Curvature, and Gradient Scale

Setup. We use the QPILOTS-M checkpoints at the end of offline training (1M steps, 4 seeds) on the five tasks of Table 8 and 256 dataset states per checkpoint, so all estimators see the same base flow and critic. From each state we integrate the unsteered base flow from noise and, at $t = 0.1, \ldots, 0.9$, draw 1024 samples from the base-flow posterior $p_{1|t}(\cdot \mid x_t, s)$ by simulating its SDE (with the σ_t of Section 4) with 20 Euler–Maruyama steps.

Critic reliability. We measure the standard deviation of the critic ensemble at the point where each estimator queries it, x_t (as in classifier guidance and QPILOTS-N) or $\hat{x}_1$ (QPILOTS-U), divided by its mean at the dataset actions of the same states. At x_t the ensemble disagrees more on every task (Table 8), most at early steps (up to $3.1\times$ at $t = 0.1$), falling to 1.1–1.5$\times$ at $t = 0.9$, whereas at $\hat{x}_1$ it stays nearly constant in t.

Critic curvature. The Jensen gap $(\mathbb{E}[\bar{Q}(s, X_1)] - \bar{Q}(s, \hat{x}_1))/\mathrm{std}[\bar{Q}(s, X_1)]$, with $X_1 \sim p_{1|t}$, measures how far $\bar{Q}$ is from linear over the posterior in units of its posterior spread. It is at most 0.11 in magnitude at every t on every task.

Gradient scale. For the tuned QPILOTS-U runs on one tuning task per domain (4 seeds), we log $\|\nabla_a \bar{Q}(s, \hat{x}_1)\|/\|v_\theta\|$ every 50K steps over offline and online training. Its per-domain median ranges from 0.024 (puzzle-3x3) to 0.73 (cube-quadruple), a $30\times$ spread, and within a run it varies by a median of $3.6\times$ (up to $25\times$ on puzzle-4x4), so a fixed unrescaled coefficient would steer with very different strength across domains and over training.

Table 8: **Critic reliability and curvature** (mean over $t = 0.1, \ldots, 0.9$ and 4 seeds). Ensemble spread relative to dataset actions, and Jensen gap in posterior standard deviations of $\bar{Q}$ (Section D.3).

Task	Spread at x_t	Spread at $\hat{x}_1$	Jensen gap
antmaze-giant t1	1.93	1.02	−0.07
cube-double t4	2.28	1.21	−0.09
cube-double-noisy t4	1.75	1.41	0.02
cube-triple t3	1.81	1.14	−0.05
humanoidmaze-large t1	1.39	1.06	0.02

D.4 Gradient Rescaling vs. σ_t Drift

Because the rescaled drift normalizes by $\|\widehat{\nabla V}_t\|$, its effective temperature varies with the state and the flow time, so the rescaled sampler does not target $\pi^\star$ for any fixed τ. We compare the rescaled drift of Eq. (8) with the unnormalized drift $\alpha_\sigma \frac{\sigma_t^2}{2} \widehat{\nabla V}_t = \alpha_\sigma \frac{1-t}{t} \widehat{\nabla V}_t$ of Section 4, both for QPILOTS-U with 4 seeds. antmaze-large task 1 and cube-double task 2 are tuning tasks. antmaze-giant task 1, cube-triple task 3, and humanoidmaze-large task 1 were selected from the QPILOTS-U results alone, for headroom below 100% and low seed variance. Table 9 lists every σ_t coefficient we ran on these tasks.

Table 9: **Rescaled vs. σ_t drift** for QPILOTS-U (final success, %, mean over 4 seeds). The rescaled column uses the per-domain α of Table 3.

Task	Rescaled (α)	σ_t drift: $\alpha_\sigma \to$ success
antmaze-large t1	98.5 (0.2)	0.2: 7.0 0.75: 44.5 1.5: 77.5 3: 95.5 6: 99.0 12: 99.5
cube-double t2	99.5 (0.3)	0.1: 99.0 0.2: 100 0.3: 100 0.4: 100 0.8: 100 1.6: 98.0
antmaze-giant t1	86.0 (0.3)	1.5: 18.0 3: 71.0 6: 90.5
cube-triple t3	77.0 (0.5)	1: 11.0 2: 0.0 4: 0.0
humanoidmaze-large t1	75.5 (0.2)	2.5: 16.0 5: 9.5 10: 11.0

The rescaled coefficient does not transfer: used as α_σ, 0.2 drops antmaze-large task 1 from 98.5% to 7%. Once retuned, σ_t steering matches or slightly exceeds rescaled steering on antmaze-large, cube-double, and antmaze-giant. It trails on cube-triple and humanoidmaze-large, where success is highest at the smallest coefficient tried (11% and 16%), so its optimum likely lies below the tested range. The unnormalized coefficient does not transfer because the gradient it multiplies has no stable scale. Along the steered trajectories of the default QPILOTS-U (median over evaluations), the magnitude-matched conversion $\alpha_\sigma/\alpha = \|v_\theta\|/(2.14\,\|\widehat{\nabla V}_t\|)$, where 2.14 is the mean of $\frac{1-t}{t}$ over the 9 steered Euler steps, ranges from 0.9 on cube-double to 14.5 on humanoidmaze-large task 4 across the eight tasks of these experiments, and $\|\widehat{\nabla V}_t\|$ changes by a median of $4.1\times$ (up to $51\times$) within a run as the critic trains. This norm includes $J_t^\top$, so it differs from the $3.6\times$ reported for $\|\nabla_a \bar{Q}\|/\|v_\theta\|$ in Section D.3.

QPILOTS-N vs. CGQL. Both evaluate the critic at the noisy x_t and use the same critic learning algorithm, but the CGQL runs come from the codebase of Li & Levine (2026), and CGQL differs from QPILOTS-N in more than the rescaling. Besides its scaling, CGQL adds $\vartheta((1-t)\tau\nabla_a Q + a_t)/t$ to the velocity, whose a_t/t term is independent of Q, and its tuned τ sits at the edge of its grid in all ten domains. Our runs do not separate these two factors.

D.5 Freezing the Base Flow Online

On OGBench the base flow keeps training with the flow-matching loss during the online phase (it is never trained against the critic). To match the VLA setting, we also freeze the base flow (and, for QPILOTS-M, the MFM) after the 10^6 offline steps and train only the critic online, with the tuned α of Table 3 and 4 seeds. Table 10 compares final success with the unfrozen runs. Averaged over the 11 cells, freezing changes success by -1.9 points for QPILOTS-M and $+2.1$ points for QPILOTS-U. The scene and cube-double cells are near ceiling in both settings and only show that freezing is harmless there. The frozen QPILOTS-M runs use $N = 16$ posterior samples, whereas the unfrozen reference uses $N = 8$.

Table 10: **Frozen vs. unfrozen base flow** during online training (final success, %, mean over 4 seeds).

	QPILOTS-M		QPILOTS-U	
Task	Frozen	Unfrozen	Frozen	Unfrozen
antmaze-large t1	91.5	98.5	89.5	99.0
antmaze-large t4	95.0	97.2	92.5	95.5
scene t2	100	100	100	100
scene t4	97.0	98.5	99.5	99.5
cube-double t2	99.5	100	100	100
cube-double t4	98.5	100	97.5	99.5
humanoidmaze-large t1	77.5	70.3	79.5	72.0
humanoidmaze-large t2	6.5	7.4	15.5	6.5
humanoidmaze-large t3	46.5	65.1	90.0	96.0
humanoidmaze-large t4	67.5	79.1	87.0	80.0
humanoidmaze-large t5	68.5	52.6	52.5	32.5

D.6 KEEPING OR DROPPING THE DENOISER JACOBIAN

QPILOTS-U differentiates the critic through the Tweedie denoiser, so its steering gradient is $J_t^\top \nabla_a \bar{Q}(s, \hat{x}_1)$ (Eq. (7)). A cheaper option uses $\nabla_a \bar{Q}(s, \hat{x}_1)$, which amounts to $J_t \approx I$, as in the concurrent QGF (Zhou et al., 2026). We test whether this choice matters. Without rescaling, dropping J_t changes the steering schedule as well as its direction; once rescaling fixes the step size, the direction alone has no consistent effect.

J_t reshapes the steering magnitude over the flow. We compute both gradients at the same states for every flow step (256 dataset states, 4 seeds, checkpoints at the end of offline training; Table 11). The critic gradient $\|\nabla_a \bar{Q}(s, \hat{x}_1)\|$ is nearly constant in t ($\approx$0.6 on humanoidmaze-large), so all of the t-dependence comes from J_t, whose mean eigenvalue grows from $\approx$0.01 to $\approx$0.9 (0.45 on cube-triple). Dropping J_t therefore enlarges the gradient $\approx$10–14$\times$ at the first steps and $\approx$1–2$\times$ at the last. The profile is stable across seeds (s.d. $\approx$0.6 or less), steering coefficients and tasks within a domain, but differs between domains, and its step average rises from about $5\times$ to $8\times$ during offline training. A constant guidance weight scales both profiles but cannot change their shape: with J_t, the steering is concentrated on the last steps. The two options differ in direction mostly early in the flow (cosine 0.13–0.15 at $t = 0.1$, above 0.9 at $t = 0.9$). The σ_t drift behaves differently again: in theory $\frac{\sigma_t^2}{2} J_t = \Sigma_t/(1-t)$, so the schedule and the Jacobian partly cancel. On humanoidmaze-large, the coefficient-free σ_t step with J_t falls from 0.59 at the first step to 0.26 and then decays slowly to 0.06, whereas without J_t it falls from 5.1 to 0.06, putting about $80\times$ more steering on the first step than on the last. Rescaling (Eq. (8)) fixes the step size at every step, so there keeping or dropping J_t changes only the direction.

Under rescaling, the direction has no consistent effect. We compare the two for QPILOTS-U with the rescaled drift and the tuned α (4 seeds; Table 12). The first six cells are tuning tasks, and the last two were selected from the QPILOTS-U results for headroom (Section D.4). There is no difference on the first six cells, four of which are at ceiling online. Offline, keeping J_t averages 52.4% vs. 48.8% over these six cells at 1M steps. Dropping J_t helps on antmaze-giant task 1 ($+8.5$ points, about 4 standard errors) and hurts on cube-triple task 3 (-19 points, about 2 standard errors). This is consistent with an ablation in QGF, in which orthogonalizing J_t, which keeps its rotation and discards its singular values, performs as well as dropping it (Zhou et al., 2026, Fig. 15).

Caveat. Early in the flow, the learned Jacobian is not the posterior covariance of Appendix B: at $t = 0.1$ its symmetric part has about 47% negative eigenvalues and it is strongly asymmetric, and both effects vanish by $t \approx 0.8$. The true J_t is close to zero there, so the network's approximation error likely dominates, in line with QGF's remark that J_t is ill-behaved at early steps. With J_t kept, the early steering direction may therefore be set largely by this error. These measurements cover two domains (humanoidmaze-large, 3 checkpoints, and cube-triple, 1 checkpoint).

Table 11: **Effect of the denoiser Jacobian by flow time** (mean over 4 seeds and 256 dataset states per checkpoint). Ratio: $\|\nabla_a \bar{Q}(s, \hat{x}_1)\| / \|J_t^\top \nabla_a \bar{Q}(s, \hat{x}_1)\|$. Cosine: between the two gradients. Eigenvalue: mean eigenvalue of the symmetric part of J_t.

	Domain	$t = 0.1$	0.3	0.5	0.7	0.9
Ratio	humanoidmaze-large	9.6	9.3	5.4	2.4	1.1
	cube-triple	12.8	12.4	9.0	5.1	1.7
Cosine	humanoidmaze-large	0.15	0.52	0.78	0.91	0.98
	cube-triple	0.13	0.35	0.52	0.68	0.93
Eigenvalue	humanoidmaze-large	0.008	0.05	0.13	0.36	0.86
	cube-triple	0.006	0.02	0.06	0.12	0.45

D.7 SENSITIVITY ANALYSIS

Fig. 6 reports the per-domain steering-coefficient sweep used to select α in Section C.4. For each OGBench domain, we run QPILOTS-M on the two tuning tasks (tasks 1 and 4 for navigation, tasks 2 and 4 for manipulation) over $\alpha \in \{0.1, 0.2, 0.3, 0.5, 1.0\}$ with 4 seeds per configuration, and pick the α with the strongest combined performance for that domain. The selected value is reused unchanged for QPILOTS-U and QPILOTS-N. On scene, cube-double,

Table 12: **Full Jacobian vs.** $J_t \approx I$ for QPILOTS-U with rescaled drift (final success, %, mean over 4 seeds).

Task	With J_t (default)	$J_t \approx I$
antmaze-large t1	98.5	98.0
antmaze-large t4	95.0	98.0
cube-double t2	99.5	100
cube-double t4	100	100
humanoidmaze-large t1	75.5	78.0
humanoidmaze-large t4	81.0	79.0
antmaze-giant t1	86.0	94.5
cube-triple t3	77.0	58.0

cube-quadruple, puzzle-3x3, and antmaze-large, success stays within 10 points of the best over a 5–10$\times$ range of α. cube-triple and humanoidmaze-large peak more sharply, at 0.3–0.5, with antmaze-giant and humanoidmaze-medium in between. The exception is puzzle-4x4-play-sparse, where success is essentially zero up to 0.5 and only emerges at $\alpha \geq 1.0$; we therefore extend the sweep to 2.0 on that domain only.

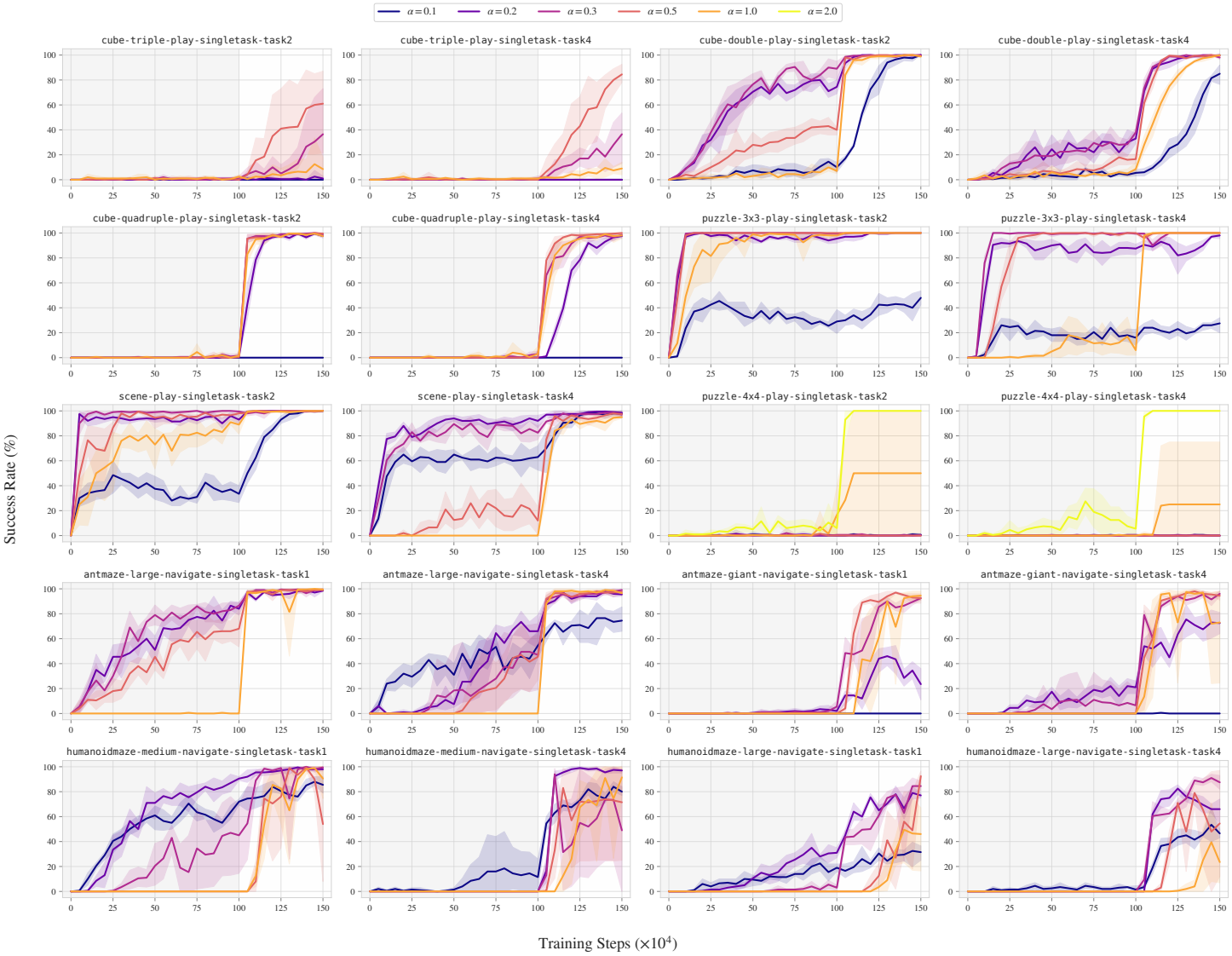


Figure 6: **Steering Coefficient α Sweep.** Eval success on the two tuning tasks of each OGBench domain across $\alpha \in \{0.1, 0.2, 0.3, 0.5, 1.0, 2.0\}$. Curves show seed-bootstrap mean and 95% CI. We only sweep 2.0 on `puzzle-4x4` since performance on others peaked well before 1.0.

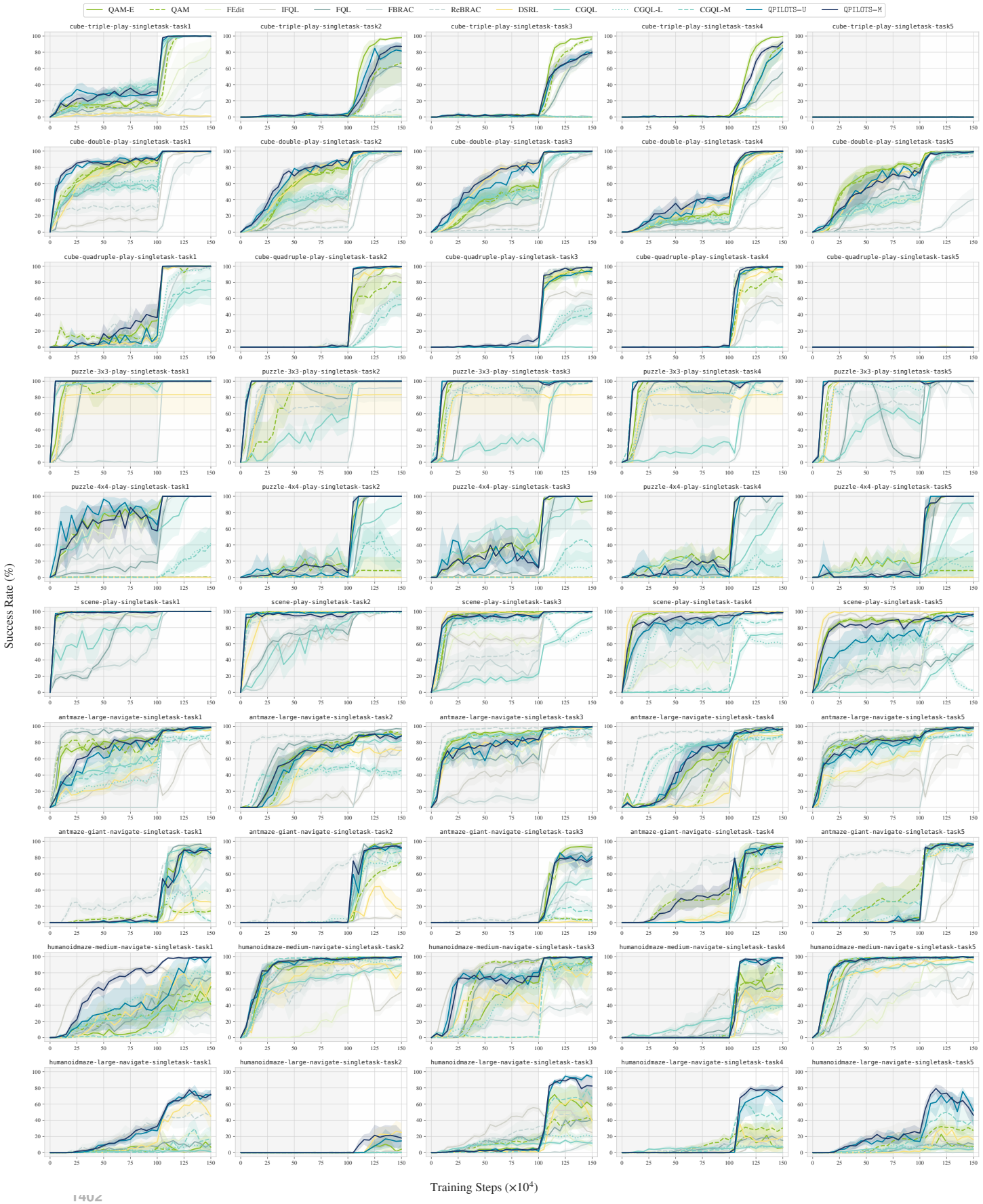
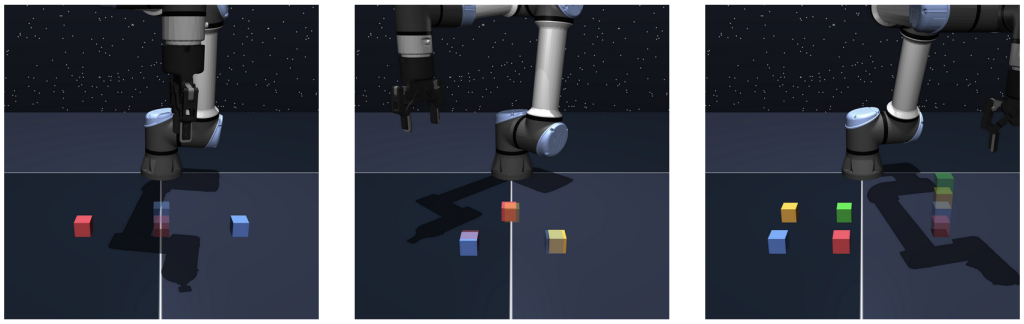


Figure 7: **OGBench: Per-Task Success Rate Across All 50 Tasks.**



cube-double cube-triple cube-quadruple



humanoidmaze-medium-navigate
Example dataset trajectory

humanoidmaze-giant-navigate
Example dataset trajectory

Figure 8: **OGBench Domains. Top:** `cube` tasks involve controlling a robot arm to achieve 5 different goal configurations per domain. **Bottom:** `humanoid` tasks involve controlling a 21-DoF humanoid to navigate to different goal locations, and is a particularly challenging long-horizon domain within the task-suite (Park et al., 2025a).

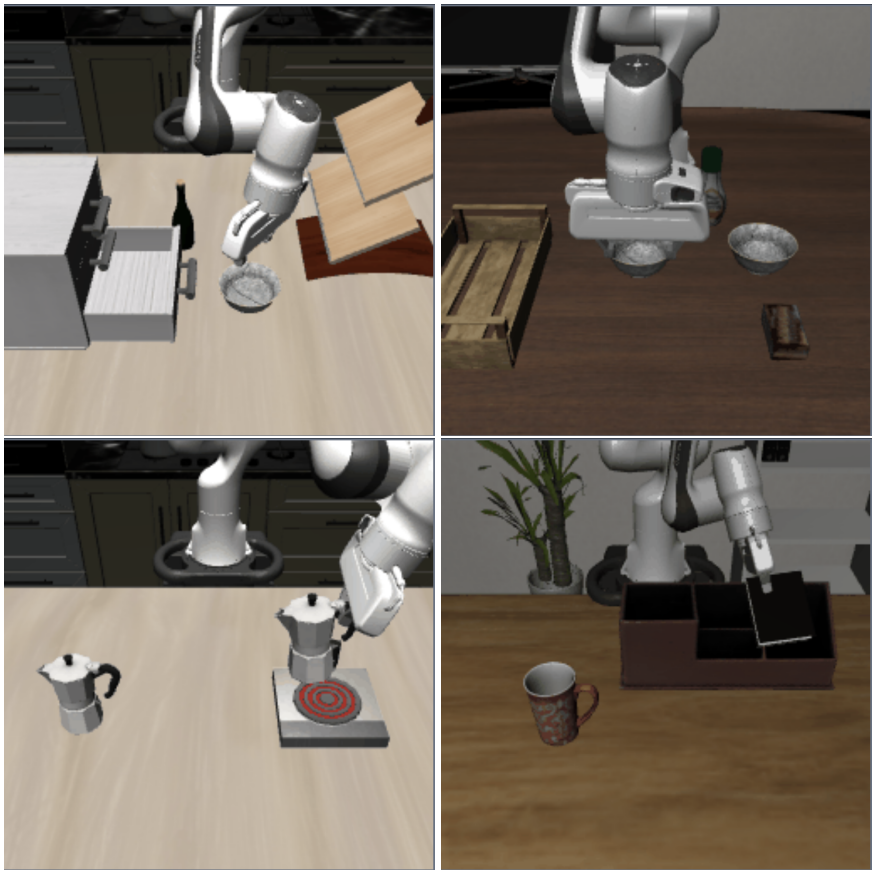


Figure 9: **LIBERO Tasks.** The LIBERO simulation task suite involves controlling a 7-DoF Franka arm to manipulate various household objects. Tasks vary in difficulty and length. (Liu et al., 2023a).