

QF3: Fast Flow RL with Filtered Q-Gradients

Chung Min Kim^{1,2}, Brent Yi^{1,2}, David McAllister^{1,2}, Hongsuk Choi^{1,2}, Himanshu Gaurav Singh¹, Jinkun Cao², Ken Goldberg¹, Pieter Abbeel^{1,2}, Carmelo Sferrazza^{2†}, Angjoo Kanazawa^{1,2†}

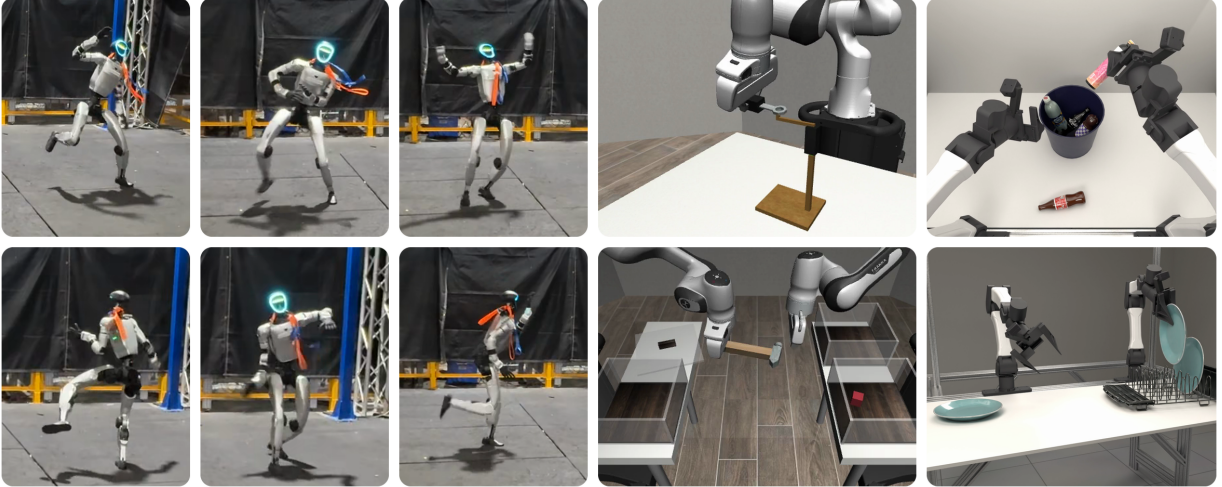


Figure 1: **QF3 is an off-policy RL algorithm for training flow policies online, either from scratch or starting from pretrained policies.** *Left:* A Unitree G1 humanoid tracking a ~ 3 minute dynamic dance motion with a whole-body motion-tracking policy trained from scratch in simulation with QF3 and deployed zero-shot on hardware. Training is an order of magnitude faster in wall-clock time than recent on-policy flow baselines. *Right:* QF3 improves manipulation performance by fine-tuning flow policies online in simulation (Robomimic tool hang and transport; ABC-sim gathering bottles, loading a dishrack).

Abstract. Flow policies have become a standard policy class for learning robot behaviors from demonstrations, but reinforcement learning is still critical for improving pre-trained flow policies or learning them from scratch through interaction. We introduce QF3 (Fast Flow RL with Filtered Q-Gradients), an online off-policy RL algorithm that trains a flow policy with flow matching plus the critic’s action gradient, backpropagated through a one-step prediction of the flow’s output. To keep updates where the critic and this prediction are reliable, QF3 applies the critic gradient only to action dimensions that stay near the replay action. To our knowledge, QF3 is the first off-policy flow RL method to train humanoid locomotion policies from scratch and transfer them zero-shot to hardware. Paired with a high-throughput off-policy training recipe, it trains humanoid locomotion and motion-tracking policies with a $10\times$ wall-clock speedup over FPO++, a recent on-policy flow RL method. We further apply QF3 to fine-tune pretrained flow-based manipulation policies on both ABC-Sim and Robomimic tasks. These results suggest that QF3 can both learn robot policies from scratch and refine those acquired from demonstrations. Website: <https://qf3-rl.github.io/>.

1 Introduction

Flow and diffusion policies are widely used in robot learning. They are the dominant approach for learning manipulation skills from demonstrations, including in large pretrained robot policies [1–5], and are increasingly adopted for locomotion and whole-body control [6, 7].

¹UC Berkeley ²Amazon FAR [†]FAR Team Co-lead Work done as an intern at Amazon FAR

Training flow policies by imitation alone is often insufficient: for some tasks, expert trajectories may be unavailable and the policy must learn from scratch through rewards; for others, a pretrained policy provides an initial behavior that can be improved through interaction, e.g., to complete a task more reliably or quickly. Settings like these have motivated a growing body of reinforcement-learning methods for flow policies, including both policy-gradient and value-based approaches [8–12], both for learning from scratch [8–11, 13] and for fine-tuning pretrained policies [12, 14, 15].

Off-policy learning is particularly attractive for robotics: not only can it reduce the need for additional interactions by reusing collected experience, but it can also support fast wall-clock training when combined with parallel simulation and large-batch updates [16]. TD3-style actors follow the critic’s action gradient $\nabla_a Q$, which gives a per-dimension direction of improvement for each action. For a flow policy, however, this means backpropagating through every step of the sampler, which is costly and can be unstable [17–20]. Existing methods for off-policy flow RL therefore either pay this cost and differentiate through the sampler [11, 17–19], freeze the flow and train a residual or noise policy on top [7, 14, 21], or use the critic only through its values or as regression targets [10, 13, 15, 22].

We introduce QF3, Fast, Filtered Q-learning for Flow policies, a general online off-policy RL algorithm for flow policies. In contrast to approaches that trains a residual policy [7, 14, 23] or modify the sampling inputs of a fixed pretrained flow [21, 24], QF3 directly trains a flow policy, even from scratch, in a stable and sample efficient manner. It is designed to drop into existing TD3-style off-policy training recipes, enabling efficient training in large-scale parallel simulation. Its actor objective is simple: it combines standard flow matching with Q -maximization through a one-step prediction of the final action, and can be evaluated directly on replay samples, without integrating or differentiating through the flow ODE.

Concretely, QF3 backpropagates a Q maximization objective through the learned velocity field, while stabilizing updates by clipping Q -induced velocity corrections around the replay-conditioned flow target. This forms a replay-local trust region: velocity coordinates whose Q -induced correction would move too far from the replay-conditioned flow target receive no critic gradient. Q -maximization therefore stays near replay actions, where the critic has been trained and the one-step prediction is accurate.

We evaluate QF3’s effectiveness both for training flow policies from scratch and for finetuning pretrained policies, on humanoid locomotion and manipulation respectively. Our contributions are as follows:

1. Algorithm and analysis. We introduce QF3, a simple off-policy actor update for flow policies, and provide an analysis of how its velocity clip filters critic-driven updates for stable off-policy learning (Sec. 3, 4). On MuJoCo, results suggest that QF3, with just a single clipped-gradient update and no sampler backpropagation, performs competitively with recent off-policy flow RL methods (Sec. 4.2).

2. Scaling to high-dimensional, dynamic control from scratch. We evaluate QF3 on velocity- and motion-tracking for the 29-DoF Unitree G1, training from scratch in simulation and deploying zero-shot on hardware; to our knowledge, this is the first off-policy flow RL method shown to do so. Our results show wall-clock training times close to FastTD3 and an order of magnitude below FPO++, an on-policy flow RL method (Sec. 5.1).

3. Manipulation fine-tuning. We apply QF3 to fine-tune pretrained flow policies in ABC-Sim [25] and Robomimic [26]. In ABC-Sim, we fine-tune the flow action head of the ABC-VLA policy with LoRA on two bimanual tasks; on gathering bottles into a bin, the fine-tuned policy completes episodes 22% faster than the base policy, and its success rate rises from 91% to 95% (Sec. 5.2). On three Robomimic tasks, where we fine-tune the full policy, results show QF3 performing similarly or better compared to OGPO [15](Sec. 5.2).

2 Related Work

2.1 Diffusion and Flow Models for Robot Policies

Diffusion and flow models—which we treat as interchangeable here, following Gao et al. [27]—have become a prominent policy class in robot learning because they model complex, high-dimensional continuous

distributions with a simple, scalable training objective. Diffusion Policy first showed that denoising models can serve as effective visuomotor policies for manipulation [28], and the idea has since scaled into generalist regimes [3, 4, 29, 30] and flow-matching VLAs that append a generative action head to a vision-language backbone [1, 2]. Generative policies are also beginning to appear in locomotion and whole-body control: BeyondMimic trains a latent diffusion policy for humanoid motion imitation with classifier guidance at test time [6], OmniXtreme pretrains a flow-matching motion prior [7], and PDP frames physics-based character animation as a diffusion-policy imitation problem [31].

Much of the success above is based on supervised learning, and optimizing these generative policies against environment rewards remains an open problem. Existing reinforcement-learning approaches have taken two main forms. One line freezes the generative prior and learns a small RL-trained policy on top—a Gaussian PPO residual [7], a latent-noise steering policy [21], an action-editing token over a frozen VLA [32], or a residual policy whose collected data is distilled back into the base VLA [23]. Another line trains the generative policy directly with on-policy policy gradients: FPO and FPO++ replace the log-likelihood in PPO’s ratio with the conditional flow-matching loss [8, 9], and ReinFlow and π_{RL} inject noise into the flow sampler to obtain tractable likelihoods [33, 34]. However, the on-policy nature of these algorithms makes them rather sample inefficient. Instead, we propose an online off-policy RL algorithm that directly trains flow policies from replay, whether learning from scratch or fine-tuning pre-trained flow policies.

2.2 Off-Policy RL with Diffusion and Flow Actors

Diffusion and flow actors have been extensively studied on simulated continuous-control benchmarks in both offline and online RL [10, 11, 17]. In offline RL, expressive generative models support value-based policy improvement while anchoring the learned policy to a fixed dataset [17, 20]. Our focus is online off-policy learning, where the actor and critic are continually updated from an evolving replay buffer.

A central design choice is how to translate critic information into updates of a generative policy. Some methods differentiate through the denoising process [17–19], while others avoid sampler backpropagation using one-step actors or supervised objectives [10, 13, 20, 35–38]. DIPO [10] and FlowDPG [12] use critic gradients to construct improved action or velocity regression targets. Recent work OGPO [15] and RL-100 [22] instead use off-policy critics in a PPO-style objective over the generative process, clipping the likelihood ratio of every denoising step as in PPO. QF3 instead follows the critic’s action gradient $\nabla_a Q$, and maximizes Q through a one-step prediction constructed from a replay action, together with standard flow matching. It clips in velocity space, rather than on a likelihood ratio, to limit the actions at which the critic is queried. This yields a simple replay-based actor update that requires no sampler backpropagation and fits directly into a TD3-style training loop.

The closest work, FlowRL [11], similarly combines Q -maximization with flow matching, but stabilizes learning by applying tanh to the flow endpoint, creating a mismatch between the flow-matching target and executed action; for pretrained policies, the added squashing also changes the action mapping at initialization, so fine-tuning no longer starts from the pretrained behavior. QF3 instead clips the Q -induced correction in velocity space around the replay-conditioned flow target, constraining critic-driven updates near replay-supported actions while retaining standard flow matching.

2.3 Stabilizing Off-Policy Q-Learning

A core challenge in off-policy Q-learning is that actor updates can chase critic overestimates, pulling the policy toward actions where the critic is unreliable. Classical actor-critic methods mitigate this with replay, target networks, and entropy regularization [39–41], while offline RL methods add explicit support constraints [42–44], critic-side conservatism [45], or behavior-regularized actor extraction [46, 47]; recent systems works make Q-learning stable at scale through wider normalization-heavy critics, target-network removal via BatchRenorm, distributional value estimates, and large-batch GPU-parallel training [16, 48–51]. QF3 integrates directly into TD3-style off-policy training. Its critic is trained as in TD3, requiring only actions

sampled from the flow and no log-likelihoods or entropy estimates, and its actor update takes the form of a TD3 actor step, backpropagating $\nabla_a Q$ into the policy through a one-step flow prediction. It therefore can leverage standard stabilization techniques and recent advances for stable Q-learning at scale.

3 Algorithm

We introduce QF3, an online off-policy algorithm for training flow-based actors. QF3 uses standard TD3-style techniques for storing transitions (s, a, r, s') in a replay buffer $\mathcal{D}$ and for training critics with Bellman backups, but represents the actor with a velocity field $v_\theta(s, x_\tau, \tau)$. At evaluation time, actions are sampled by drawing $x_0 = \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$, and integrating $dx_\tau/d\tau = v_\theta(s, x_\tau, \tau)$ from $\tau = 0$ to $\tau = 1$ using N Euler steps.

QF3 aims to steer a flow actor toward higher-value actions with the critic gradient $\nabla_a Q$, which cheaply provides an improvement direction in each action dimension. However, the exact gradient of Q with respect to the flow parameters runs through every integration step of the sampler, and backpropagating through all of them is costly and can be unstable [17–20]. We therefore approximate the final action with a single Euler step from a noised replay action and maximize Q at that prediction, while flow matching on replay actions keeps the policy close to the data. Because this update can push the prediction away from the actions that the critic is trained on, we clip the prediction in velocity space so that the critic is queried only near replay actions. We first present the actor objective, then explain how this velocity-space trust region stabilizes policy improvement.

3.1 QF3 Actor Objective

The QF3 actor objective consists of two terms: a conditional flow-matching (CFM) term that anchors the actor to replay-buffer actions and a critic term for policy improvement.

Anchoring the flow with the replay buffer. We construct the CFM term from a replay transition $(s, a_{\text{buf}}, r, s')$ by sampling $\epsilon \sim \mathcal{N}(0, I)$ and $\tau \sim \mathcal{U}(0, 1)$ and forming

$$x_\tau = (1 - \tau)\epsilon + \tau a_{\text{buf}}, \quad u_{\text{buf}} = a_{\text{buf}} - \epsilon$$

where u_{buf} is the conditional velocity that transports ϵ to a_{buf} along the linear path. We supervise the predicted velocity with the standard CFM loss:

$$\mathcal{L}_{\text{cfm}} = \|v_\theta(s, x_\tau, \tau) - u_{\text{buf}}\|^2 \quad (1)$$

On its own, this flow-matching loss trains the actor to reproduce the replay distribution, that is, it regularizes the actor toward the behavior policy that collected the data, which the replay buffer approximates non-parametrically, as in behavior-regularized actor updates [46]. By regularizing towards this behavior policy, we also keep the actor close to the actions for which the critic’s estimates are most reliable. The policy improves over time as the actor uses the critic gradients to steer its actions towards higher-value actions, and learns to flow towards these improved actions, which are placed into an evolving replay buffer.

Critic-driven policy improvement. The core contribution of QF3 is how critic gradients reach the velocity field. Directly maximizing the critic Q through a one-step prediction of the action is simple and cheap, but can be unstable. This instability does not come from the sampler, since the update never backpropagates through it, but from where the critic is queried. The critic is evaluated at the one-step prediction, which can drift away from replay actions, and the critic’s estimates can become unreliable. Integrating the flow ODE from x_τ with a single Euler step gives a one-step approximation of the action the policy would produce, and we maximize the critic through it:

$$\begin{aligned} \hat{x}_1 &= x_\tau + (1 - \tau)v_\theta(s, x_\tau, \tau), \\ \mathcal{L}_{\text{direct}} &= -Q(s, \hat{x}_1) + \lambda_{\text{cfm}}\mathcal{L}_{\text{cfm}}. \end{aligned} \quad (2)$$

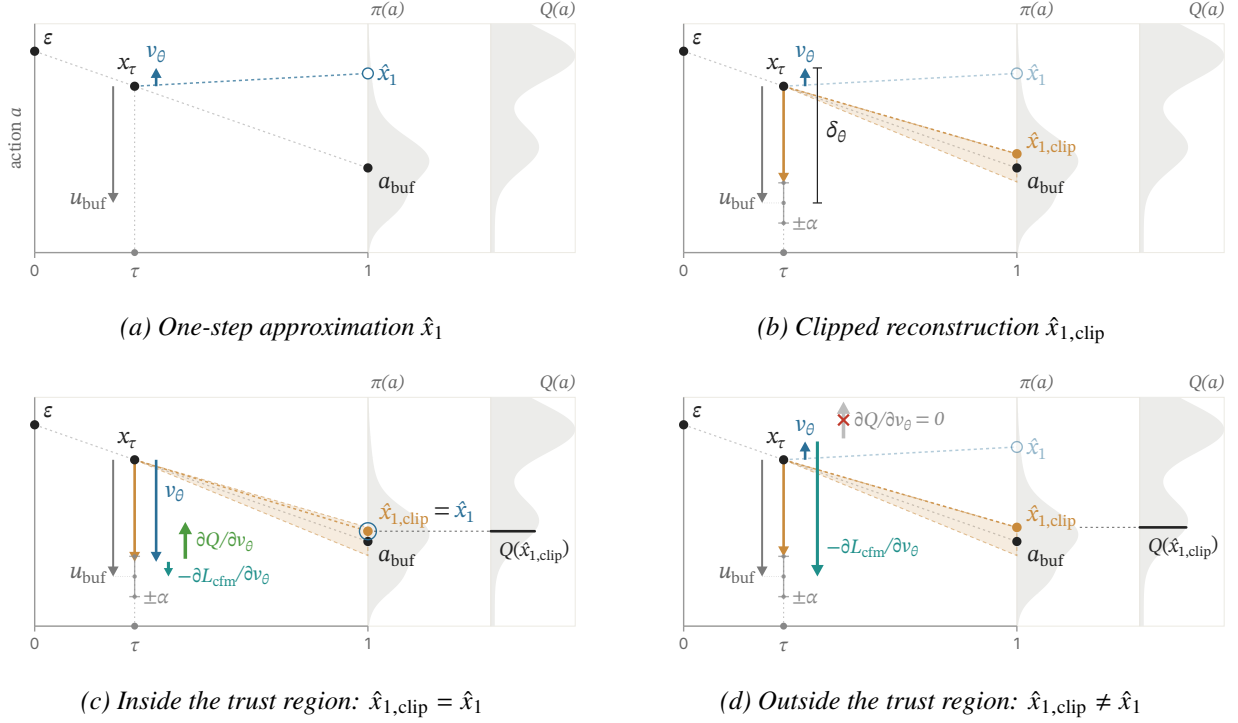


Figure 2: **The QF3 actor update in 1D.** A one-step prediction from a noised replay action can land far from the replay action, where the critic’s estimates are unreliable (a). Clipping the velocity keeps the critic’s query near the replay action (b). Inside this trust region, the critic gradient improves the policy (c); outside it, the critic gradient is dropped and only the flow-matching loss on the replay action acts, pulling the policy back toward the replay action (d).

Like a TD3 actor, this update backpropagates $\nabla_a Q$ into the policy, but through a single velocity evaluation. The one-step approximation is exact when $v_\theta = u_{\text{buf}}$, which gives $\hat{x}_1 = a_{\text{buf}}$. However, in online training, a randomly initialized actor can push $\hat{x}_1$ off the critic’s support (Fig. 2a), where it can chase overestimated Q -values and the velocity magnitudes diverge.

To address this instability, we introduce a clipped actor objective that induces a local trust region in velocity space. By clipping the policy’s velocity around the replay-conditioned flow target, we keep the action evaluated by the critic close to the replay action, where both the one-step approximation and the critic remain valid. Let $\delta_\theta := v_\theta(s, x_\tau, \tau) - u_{\text{buf}}$ be the policy-buffer gap, the difference between the predicted and replay-conditioned velocities. We clip it coordinate-wise to $[-\alpha, \alpha]$ and evaluate the critic at the corresponding one-step action:

$$\begin{aligned} \hat{v}_{\theta,\text{clip}} &= u_{\text{buf}} + \text{clip}(\delta_\theta, -\alpha, +\alpha), \\ \hat{x}_{1,\text{clip}} &= x_\tau + (1 - \tau)\hat{v}_{\theta,\text{clip}} \\ &= a_{\text{buf}} + (1 - \tau)\text{clip}(\delta_\theta, -\alpha, +\alpha), \end{aligned} \quad (3)$$

where the last equality uses $x_\tau + (1 - \tau)u_{\text{buf}} = a_{\text{buf}}$, so the critic is evaluated only within $(1 - \tau)\alpha$ of the replay action in each coordinate (shaded wedge in Fig. 2b). Finally, QF3 minimizes

$$\mathcal{L}_{\text{actor}} := -Q(s, \hat{x}_{1,\text{clip}}) + \lambda_{\text{cfm}}\mathcal{L}_{\text{cfm}}. \quad (4)$$

This separation is important: when the policy-buffer gap is large, clipping prevents the resulting endpoint from moving arbitrarily far from the replay action in the Q term, while the flow-matching loss still penalizes

the full, unclipped gap.

3.2 Filtered Q-Gradients as a Trust Region

Clipping not only bounds the one-step reconstruction, but also filters the critic gradients that reach the actor. Differentiating the Q term in Eq. 4 through the clipped reconstruction in Eq. 3 gives

$$\begin{aligned}\frac{\partial Q}{\partial v_\theta} &= (1 - \tau) m \odot \nabla_a Q(s, a)|_{a=\hat{x}_{1,\text{clip}}}, \\ m_i &= \mathbb{1}[|\delta_{\theta,i}| < \alpha]\end{aligned}\tag{5}$$

For each replay sample, velocity component i receives a Q -gradient only when its policy-buffer gap lies within the trust-region bounds (Fig. 2c). When that component saturates the clip, its Q -gradient is masked out, while the CFM term remains active and pulls the raw velocity prediction toward u_{buf} (Fig. 2d). If the gap re-enters the clipping range, the Q -gradient resumes.

The trust region thus acts as a policy-consistency filter: critic gradients drive local improvement within the allowed velocity range, while flow matching pulls predictions outside that range back toward the replay-conditioned target. This is analogous to the ratio clip of PPO [52], which likewise enforces a trust region by withholding the policy-improvement gradient. We write an interpretation of this analogy through the CFM–ELBO correspondence of Kingma and Gao [53], reading the clip as a trust region on an ELBO surrogate of the replay action, and sketch this out in Appendix A.2.

4 Analyzing Filtered Q-Gradients

4.1 Analyzing the velocity clip

In this section, we analyze the QF3’s actor update described in Sec. 3 within Gymnasium [54] MuJoCo-v4. Results are plotted in Figure 3. We find that:

The velocity-space clipping in the actor loss is essential for stable training. The clip in QF3’s actor update (Eq. 3) prevents large, destabilizing updates from the critic early in training when the value estimates are unreliable, which in turn spikes the magnitude across the velocity field, debilitating learning. We verify this claim on Humanoid-v4 against *Direct Q (unclamped)*, an ablation that applies the critic’s gradient to the actor without the per-element velocity clip (Fig. 3a), and find that the unclamped update collapses learning even at a 10× lower actor learning rate (3e-5 instead of the default 3e-4), while the default QF3 clip recovers a usable policy. A similar collapse seems to reappear when finetuning a pretrained manipulation policy (Sec. 5.2), where the unclamped update drops a policy that already solves the task to near-zero success within the first 100k steps before recovering.

Velocity-space clipping creates a lower-curvature flow field than action-space clipping. The velocity-space clip in QF3 effectively induces a τ -dependent action-space trust region of size $(1 - \tau)\alpha$ (see Fig. 2b). Here, we ablate this decision by clipping the correction from $\hat{x}_1$ instead, effectively making α time-dependent ($\frac{\alpha}{1-\tau}$; Fig. 3b). We find that although this may lead to a similar policy performance, the learned flow field may have a much higher curvature (Fig. 3c). The reason is that a fixed action-space bound α translates to a velocity bound of $\alpha/(1 - \tau)$, which is unbounded as $\tau \rightarrow 1$: the critic can inject arbitrarily large velocity corrections exactly where the field determines the final action, and the learned field bends sharply to accommodate them.

The velocity-space clip remains active throughout training. Since the *Direct Q (unclamped)* collapse occurs early in training, one may expect the clip to have no effect once the critic is reliable and the policy is stable. We test this by running QF3 on Ant and Humanoid with $\alpha = \frac{1}{2}$, and logging the fraction of action dimensions whose correction lies within the clip, $|\delta_{\theta,i}| < \alpha$, and hence receives a Q -gradient (Eq. 5). We find that the clipped minority shrinks as the critic improves but does not vanish, so the clip continues to filter critic corrections even after the policy is performant: at 50k, 250k, and 500k steps, this fraction is 74/83/85% on

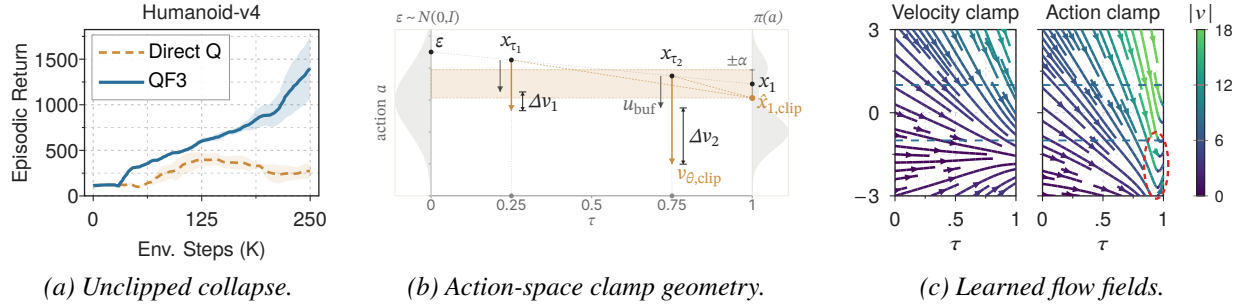


Figure 3: **Analyzing the velocity clip.** (a) Without the clip, Direct Q collapses on Humanoid-v4 even at a 10 $\times$ lower actor learning rate. (b) Clamping the critic input to a fixed band around a_{buf} instead allows a policy-buffer gap of up to $\alpha/(1 - \tau)$, without bound as $\tau \rightarrow 1$. (c) On HalfCheetah-v4, the learned flow field then bends sharply near $\tau = 1$ (circled), which the velocity clip avoids.

Ant and 81/83/82% on Humanoid. This suggests that the clip is therefore not only a safeguard against an unreliable early critic, but may act as a trust region that constrains the actor update for the entirety of training.

4.2 Validation on MuJoCo-v4

We use the same Gymnasium MuJoCo-v4 environments to compare QF3 against recent off-policy flow-based RL methods, where all share the same TD3 harness, flow actor, and critic (Appendix A.3). This comparison isolates how they inject critic information into the flow actor: (i) as a *regression target*, where QSM [13] directly aligns v_θ with $\nabla_a Q$; (ii) as a *regularized objective*, where FlowRL [11] pairs Q -maximization with a Wasserstein-2 penalty; and (iii) as *improved targets*, where DIPO [10] improves replay actions under the critic and refits the actor to them. QF3 instead uses the pathwise Q gradient $\nabla_a Q$ to update the actor.

We evaluate on Hopper, Walker2d, Ant, and Humanoid (Fig. 4). Every flow method is tuned with the same protocol: we sweep its method-specific hyperparameters (Table A.2) and select the best configuration per task by mean return over steps 450k–500k and seeds. QF3 performs competitively with one velocity evaluation per actor update, whereas DIPO takes 10–40 action-gradient steps and FlowRL trains an extra expected value function. QSM fails on Ant and lags far behind on Walker2d for every score-loss weight in its sweep (Fig. A.1a), suggesting that matching the gradient’s scale is harder than following its direction. A single QF3 setting ($\alpha = 1$, $\lambda_{\text{cfm}} = 0.3$) reaches at least 79% of its per-task best return on all four tasks (Fig. A.1b).

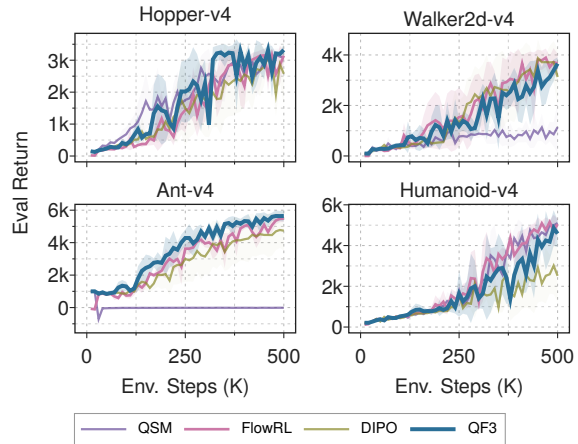


Figure 4: **QF3 in MuJoCo-v4 [54].** The plots show evaluation returns over 500k environment steps (mean $\pm$ std across 3 seeds). QF3 performs competitively with state-of-the-art flow RL methods.

5 Experimental Results

We evaluate QF3 in the two regimes: first, we scale to humanoid control, combining the actor update with the high-throughput training recipe of FastTD3 [16]. This allows QF3 to train velocity- and motion-tracking policies in Holosoma [55] at wall-clock speeds much faster than the on-policy flow baseline FPO++ [9]—the first off-policy flow RL method, to our knowledge, to train a humanoid policy from scratch and transfer

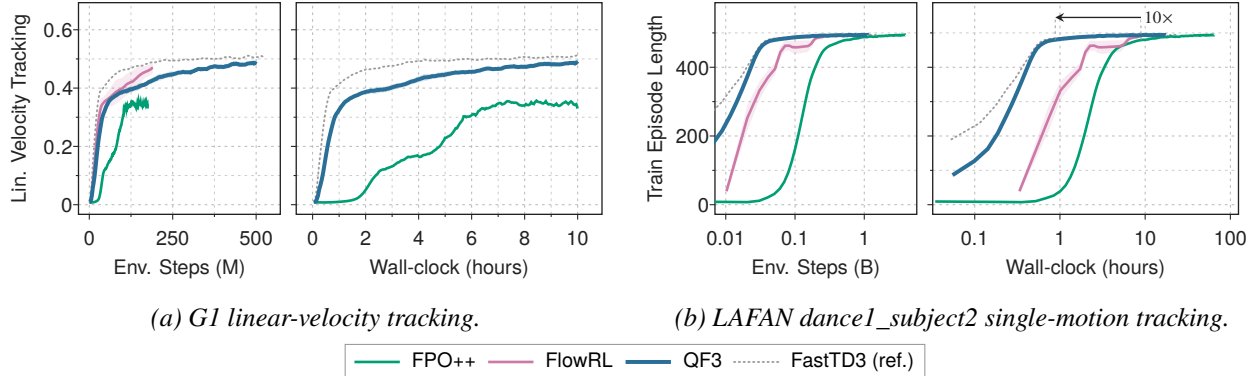


Figure 5: **Humanoid locomotion learning curves.** Each panel shows env.-step (left) and wall-clock (right) views of the same task. (a) Per-step `raw_lin_vel` reward under a fixed 10-hour budget on a single A10G. (b) Mean episode length (max 500) on LAFAN dance motion, in log scale. QF3 converges $\approx 10\times$ faster in wall-clock than FPO++ [9]. FlowRL’s velocity-tracking runs end at 190M env steps.

zero-shot to hardware (Sec. 5.1, Fig. 1). Second, we fine-tune pretrained manipulation policies in ABC Sim, where QF3 trains LoRA adapters on a VLA’s flow head (Sec. 5.2), and on Robomimic, where it updates the full policy (Sec. 5.2).

5.1 Evaluating QF3 for humanoid locomotion

Flow policies are increasingly used as pretrained priors for humanoid whole-body control [6, 7], but improving them with RL still relies on residual policies over a frozen prior [7]. As a first step toward native flow RL in this setting, we ask whether QF3 can train a flow policy from scratch within FastTD3’s massively parallel, high-throughput recipe as efficiently as FastTD3 itself.

To study this, we train policies using Holosoma [55], an open-source humanoid whole-body RL framework that supplies both the task suite (velocity tracking, single-motion tracking) for the 29-DoF Unitree G1 and a tuned FastTD3 [16] baseline, and compare against FPO++ by porting the mixed rough terrain environment from Holosoma into their open-source code release. FastTD3 is a high-throughput off-policy recipe that pairs TD3’s clipped double-Q critic and target networks with massively parallel GPU simulation, a large replay buffer, large-batch updates, and a distributional categorical critic [56]. Because QF3 keeps TD3’s replay buffer, critic, and target-update structure (Sec. 3) and only changes the actor, this recipe drops in unchanged: we replace the deterministic FastTD3 actor with the QF3 flow actor and its clipped actor loss (Eq. 3), use the mean of the categorical critic as the scalar $Q(s, \hat{x}_1)$ in the actor update, and share all remaining hyperparameters with FastTD3 (Appendix A.4). At deployment, the flow is integrated deterministically from $\epsilon = 0$ with 5 Euler steps, following FPO++ [9].

We report learning curves and wall-clock time comparisons in Fig. 5, with mean $\pm$ std over 3 seeds; each run trains on a single GPU.¹ Results show that QF3 succeeds in both the humanoid locomotion and motion tracking environments. QF3 trains an order of magnitude faster than the on-policy baseline FPO++ [9], and is competitive with Holosoma’s optimized FastTD3 implementation. On velocity tracking, QF3 finishes within $\sim 4\%$ of FastTD3’s final tracking reward under the same 10-hour budget. On motion tracking, QF3 reaches full-length (500-step) episodes in roughly half an hour of wall-clock, close to FastTD3, while FPO++ takes $\approx 10\times$ longer in both wall-clock and environment steps. FlowRL [11], another off-policy flow method that we run on the same FastTD3 recipe, reaches comparable tracking performance on both tasks but is less consistent: its seeds vary about $3\times$ as much as QF3’s on velocity tracking and $9\times$ on motion tracking, perhaps

¹Velocity-tracking runs use an NVIDIA A10G and motion-tracking runs an NVIDIA L40S. FlowRL’s velocity-tracking runs used an NVIDIA L4, so we omit FlowRL from the velocity wall-clock panel.

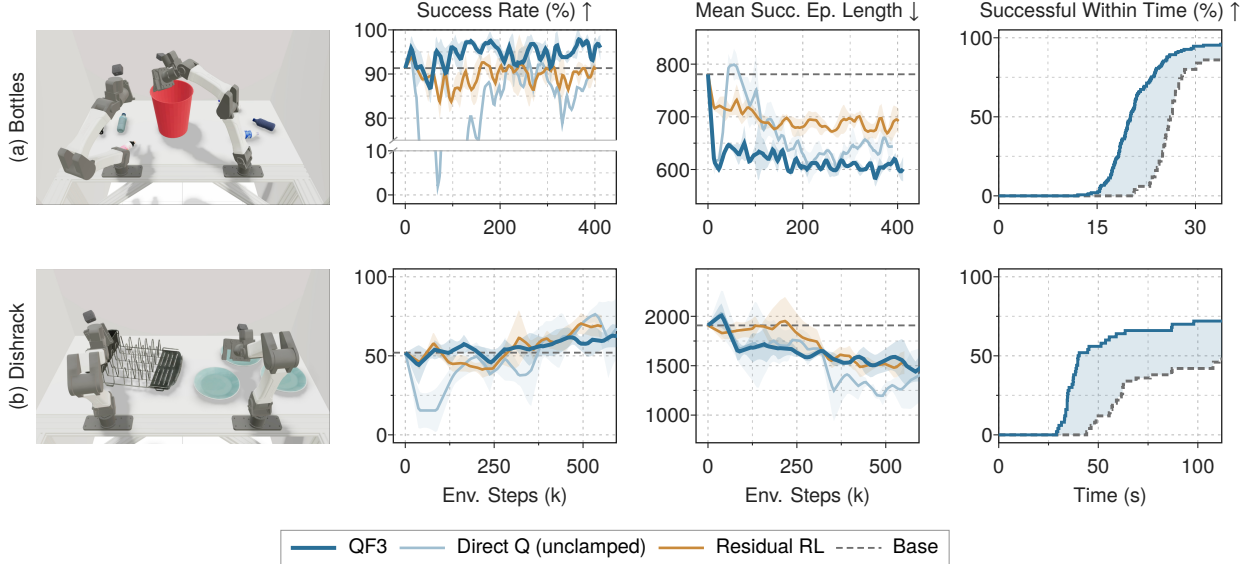


Figure 6: **ABC-VLA manipulation fine-tuning.** (a) *Throw Bottles in Bin* and (b) *Load Plates in Dishrack*, 3 seeds, mean $\pm$ std. The right column shows the share of episodes solved within each time for the pretrained base policy and the final checkpoint of one QF3 run. The shaded gap is the drop in mean episode time, counting failures at the limit (5.8 s, 27 s).

due to the additional expectile value function it trains to regularize its actor. On motion tracking it also needs about $1.8\times$ as many environment steps as QF3 to reach near-full-length (490-step) episodes.

We also find policies trained with QF3 can be successfully deployed on real robots zero-shot, with no real-world fine-tuning and the same joint-target action space and PD gains as in simulation: we find that the G1 humanoid policies trained using QF3 can follow velocity commands and track long, challenging, and dynamic motions (LAFAN “dance1_subject2”, approx. 3 minutes; see Fig. 1).

5.2 Fine-tuning pretrained manipulation policies

We next ask whether the same core update can improve a pretrained generative policy rather than train one from scratch. In both settings below, we add a flow-matching loss toward the frozen base policy, $\lambda_{\text{base}} \|v_{\theta} - v_{\text{base}}\|^2$, to the QF3 actor loss, and find this anchor important for stable fine-tuning [57].

ABC-Sim We fine-tune the vision-conditioned ABC-VLA policy [25] in ABC Sim on two bimanual tasks, *Throw Bottles in Bin* and *Load Plates in Dishrack* (Fig. 6). The same pretrained policy is used for both tasks, and pairs a frozen VLM with a DiT action head that integrates a 5-step flow into a 30-step action chunk; QF3 trains rank-4 LoRA adapters [58] on the attention and MLP projections of the head, initialized to zero so that training starts exactly at the base policy. We compare against residual RL, which freezes the policy and trains a bounded additive correction with TD3 [14], sharing the same critic, replay buffer, simulator, and evaluation protocol, and against *Direct Q (unclamped)*, which is QF3 without the velocity field trust-region. An episode succeeds when all five bottles are in the bin or all three plates are in the rack before the timeout (1000 and 3300 control steps, or 34 and 112 s). *Bottles* rewards +1 for each bottle the first time it enters the bin. *Dishrack* rewards progress normalized by object count (+1/3 per newly racked plate) plus a +1 bonus on success. Both tasks run at a 29.4 Hz control rate, and the policy executes the first 15 steps of each chunk, about two policy calls per second. Training collects experience in 16 parallel simulated worlds (Appendix A.5, Table A.3).

We report success rate and the mean length of successful episodes from 50-episode evaluations in Fig. 6. QF3 improves the pretrained policy on both tasks. On *Bottles*, successful episodes finish 22% faster (781 $\rightarrow$ 606 steps) and success rises from 91% to 95%; over the final 100k steps this is 12% faster and 5 points

higher than residual RL. The speedup is spread evenly over this task: each of the five bottles takes about 1 s (29 steps) less. On *Dishrack*, QF3 improves both success rate and successful-episode length relative to the pretrained base policy, with performance comparable to residual RL. Here the gain comes mostly from avoiding failures, through two behavior changes (App. A.7). On the first plate, the base policy usually grasps with the left hand, then the right, then the left again; QF3 instead settles on a single right-to-left handover (which the base policy already produces occasionally), dropping the extra grasps. It also seats plates more upright (9° vs. 14° median tilt), so only 8% of placed plates fall back out of the rack, against 21% for the base policy. *Direct Q (unclamped)* suffers a pronounced transient collapse on both tasks before recovering, consistent with the instability observed in Sec. 4.

Robomimic We also show full finetuning of pretrained flow policies with QF3 on three Robomimic tasks [26], *Square*, *Tool Hang*, and *Transport* (Fig. 7). We compare against OGPO [15], which fine-tunes the full policy with PPO-style updates over its denoising steps using the critic’s value as the reward, and against DSRL [21] and EXPO [14], which do not fine-tune the base policy with RL but steer its input noise and edit its actions, respectively. We follow OGPO’s online-only protocol in its released codebase: all methods start from the same behavior-cloned flow policy (stopped at $\leq 50\%$ success on *Square* and *Tool Hang*) and use no demonstrations online. QF3 replaces only OGPO’s actor update, and QF3+ adds OGPO’s success-buffer regularizer, a flow-matching loss on actions from successful episodes. For OGPO we run OGPO+CA, the variant its authors recommend, which uses the same regularizer and additionally applies conservative advantages, updating on an action only when every critic in the ensemble agrees on the sign of its advantage (Appendix A.6). Following OGPO, QF3 and QF3+ act by sampling 8 candidate actions and executing the one the critic scores highest.

Results in Figure. 7 show QF3+ matches or exceeds OGPO+CA in final success rates over the three tasks. We also find that QF3 does not need the success-buffer regularizer that OGPO relies on: without it, QF3 reaches 96% success on *Tool Hang* and 96% on *Transport*, whereas OGPO without it saturates near 80% and 40% on these tasks in its authors’ experiments [15, Fig. 9]. Because QF3’s actor follows the critic’s gradient directly, it is more sensitive to an untrained critic than OGPO’s clipped policy-gradient update, so we train the critic alone on base-policy rollouts before the first actor update (20k steps on *Square* and *Tool Hang*, 100k on *Transport*). We find that DSRL and EXPO, which leave the base policy’s weights unchanged, struggle to improve performance on the long-horizon tasks.

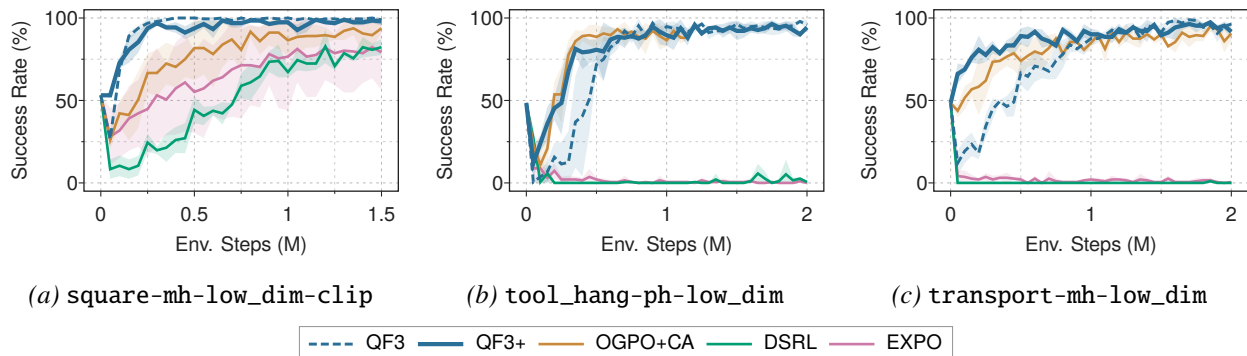


Figure 7: **Robomimic fine-tuning.** Success rate versus online environment steps on three Robomimic tasks. All methods start from the same behavior-cloned checkpoint (step 0) and are evaluated with the policy they act with: QF3, QF3+, and OGPO+CA with SDE sampling, DSRL with its noise policy, and EXPO with its edited actions. 3 seeds, mean $\pm$ std.

6 Conclusion

We present QF3, an online off-policy reinforcement-learning algorithm for flow policies. QF3 trains a flow actor directly from replay by combining standard flow matching with critic-driven improvement through a one-step prediction of the denoised action. To stabilize this update, QF3 clips critic-induced velocity corrections around the replay-conditioned flow target. This bounds where the critic is queried and masks Q-gradients when the actor deviates too far from replay, providing a simple policy-consistency filter for off-policy learning. The actor update requires neither integration nor differentiation through the flow ODE. On continuous-control benchmarks, QF3 performs competitively with recent off-policy flow RL methods. Paired with the FastTD3 training recipe, it trains Unitree G1 velocity- and motion-tracking policies at wall-clock speeds close to FastTD3 and approximately an order of magnitude faster than FPO++, with zero-shot transfer to hardware. The same update also fine-tunes a pretrained ABC-VLA policy on two bimanual manipulation tasks in simulation, improving success rate and successful-episode duration over the base policy. On three Robomimic tasks, it finetunes the full policy to over 95% success. Together, these results suggest that a single replay-based flow actor update can support both learning robot behaviors from scratch and improving flow policies acquired from demonstrations.

Acknowledgments

This project was funded in part by NSF:CNS-2235013, IARPA DOI/IBC No. 140D0423C0035, DARPA No. HR001123C0021, BAIR Commons sponsors, and the Bakar Fellows Program. Chung Min Kim and Brent Yi are supported by the NSF Graduate Research Fellowship Program, Grant DGE 2146752. We thank Qiyang Li, Zhiyuan Zhou, and Arthur Allshire for fruitful technical conversations.

References

- [1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, “ π_0 : A vision-language-action flow model for general robot control,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2025. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [2] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, “ $\pi_{0.5}$: a vision-language-action model with open-world generalization,” *arXiv preprint arXiv:2504.16054*, 2025. [Online]. Available: <https://arxiv.org/abs/2504.16054>
- [3] TRI LBM Team, “A careful examination of large behavior models for multitask dexterous manipulation,” *arXiv preprint arXiv:2507.05331*, 2025. [Online]. Available: <https://arxiv.org/abs/2507.05331>
- [4] NVIDIA, “GR00T N1: An open foundation model for generalist humanoid robots,” *arXiv preprint arXiv:2503.14734*, 2025. [Online]. Available: <https://arxiv.org/abs/2503.14734>
- [5] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjorck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin, L. Magne, A. Mandlekar, A. Narayan, Y. L. Tan, G. Wang, J. Wang, Q. Wang, Y. Xu, X. Zeng, K. Zheng, R. Zheng, M.-Y. Liu, L. Zettlemoyer, D. Fox, J. Kautz, S. Reed, Y. Zhu, and L. Fan, “DreamGen: Unlocking generalization in robot learning through video world models,” *arXiv preprint arXiv:2505.12705*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.12705>

- [6] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu, “BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion,” *arXiv preprint arXiv:2508.08241*, 2025. [Online]. Available: <https://arxiv.org/abs/2508.08241>
- [7] Y. Wang, S. Zhu, P. Zhi, Y. Li, J. Li, Y.-L. Li, Y. Xiao, X. Wang, B. Jia, and S. Huang, “OmniXtreme: Breaking the generality barrier in high-dynamic humanoid control,” *arXiv preprint arXiv:2602.23843*, 2026. [Online]. Available: <https://arxiv.org/abs/2602.23843>
- [8] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa, “Flow matching policy gradients,” *arXiv preprint arXiv:2507.21053*, 2025. [Online]. Available: <https://arxiv.org/abs/2507.21053>
- [9] B. Yi, H. Choi, H. G. Singh, X. Huang, T. E. Truong, C. Sferrazza, Y. Ma, R. Duan, P. Abbeel, G. Shi *et al.*, “Flow policy gradients for robot control,” *arXiv preprint arXiv:2602.02481*, 2026.
- [10] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin, “Policy representation via diffusion probability model for reinforcement learning,” *arXiv preprint arXiv:2305.13122*, 2023. [Online]. Available: <https://arxiv.org/abs/2305.13122>
- [11] L. Lv, Y. Li, Y. Luo, F. Sun, T. Kong, J. Xu, and X. Ma, “Flow-based policy for online reinforcement learning,” in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. [Online]. Available: <https://openreview.net/forum?id=CANUXhPoyn>
- [12] K. Shi, J. Shi, P. Hebbbar, Z. Zhao, T. Amarnath, Y. Su, S. Bahl, and D. Pathak, “FlowDPG: Deterministic policy gradient on flow matching policies for real-world manipulation,” *arXiv preprint arXiv:2606.22303*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.22303>
- [13] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma, “Learning a diffusion model policy from rewards via q-score matching,” in *Proceedings of the 41st International Conference on Machine Learning*, 2024.
- [14] P. Dong, Q. Li, D. Sadigh, and C. Finn, “Expo: Stable reinforcement learning with expressive policies,” *arXiv preprint arXiv:2507.07986*, 2025.
- [15] S. Patil, M. Nakamoto, M. Agarwal, S. Saxena, J. Zhang, G. Anantharaman, C. Winston, C. Pan, D. Chen, N.-C. Huang, Z. Temel, O. Kroemer, S. Levine, A. Gupta, H. Dai, P. Shah, and M. Simchowitz, “OGPO: Sample efficient full-finetuning of generative control policies,” *arXiv preprint arXiv:2605.03065*, 2026. [Online]. Available: <https://arxiv.org/abs/2605.03065>
- [16] Y. Seo, C. Sferrazza, H. Geng, M. Nauman, Z.-H. Yin, and P. Abbeel, “FastTD3: Simple, fast, and capable reinforcement learning for humanoid control,” *arXiv preprint arXiv:2505.22642*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.22642>
- [17] Z. Wang, J. J. Hunt, and M. Zhou, “Diffusion policies as an expressive policy class for offline reinforcement learning,” in *International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=AHvFDPi-FA>
- [18] Y. Wang, L. Wang, Y. Jiang, W. Zou, T. Liu, X. Song, W. Wang, L. Xiao, J. Wu, J. Duan, and S. E. Li, “Diffusion actor-critic with entropy regulator,” in *Advances in Neural Information Processing Systems*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.15177>
- [19] Y. Zhang, S. Yu, T. Zhang *et al.*, “SAC Flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling,” *arXiv preprint arXiv:2509.25756*, 2025.

- [20] S. Park, Q. Li, and S. Levine, “Flow q-learning,” in *International Conference on Machine Learning (ICML)*, 2025.
- [21] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, “Steering your diffusion policy with latent space reinforcement learning,” *arXiv preprint arXiv:2506.15799*, 2025.
- [22] K. Lei, H. Li, D. Yu *et al.*, “RL-100: Performant robotic manipulation with real-world reinforcement learning,” *arXiv preprint arXiv:2510.14830*, 2025.
- [23] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, F. Hu, J. Wu, Z. Luo, L. Fan *et al.*, “Self-improving vision-language-action models with data generation via residual rl,” *arXiv preprint arXiv:2511.00091*, 2025.
- [24] Z. Xu, R. Gong, M. V. Minniti, A. S. Gundogdu, E. Rosen, K. Sivakumar, R. Yan, Z. Wang, D. Deng, P. Stone *et al.*, “Expertgen: Scalable sim-to-real expert policy learning from imperfect behavior priors,” *arXiv preprint arXiv:2603.15956*, 2026.
- [25] A. Allshire, H. G. Singh, R. Singh, A. Rashid, H. Choi, D. McAllister, J. Yu, Y. Chen, H. Huang, P. Abbeel, X. Chen, R. Duan, P. Isola, J. Malik, F. Shentu, G. Shi, P. Wu, and A. Kanazawa, “Scalable behavior cloning with open data, training, and evaluation,” *arXiv preprint arxiv:2606.27375*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.27375>
- [26] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, “What matters in learning from offline human demonstrations for robot manipulation,” in *Conference on Robot Learning (CoRL)*, 2021. [Online]. Available: <https://arxiv.org/abs/2108.03298>
- [27] R. Gao, E. Hoogeboom, J. Heek, V. De Bortoli, K. P. Murphy, and T. Salimans, “Diffusion models and gaussian flow matching: Two sides of the same coin,” in *The Fourth Blogpost Track at ICLR 2025*, 2025.
- [28] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.04137>
- [29] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, “Octo: An open-source generalist robot policy,” *arXiv preprint arXiv:2405.12213*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.12213>
- [30] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, “RDT-1B: a diffusion foundation model for bimanual manipulation,” *arXiv preprint arXiv:2410.07864*, 2024. [Online]. Available: <https://arxiv.org/abs/2410.07864>
- [31] T. E. Truong, M. Pisen, Z. Xie, and K. Liu, “Pdp: Physics-based character animation via diffusion policy,” in *SIGGRAPH Asia 2024 Conference Papers*, 2024, pp. 1–10.
- [32] C. Xu, J. T. Springenberg, M. Equi, A. Amin, A. Esmail, S. Levine, and L. Ke, “Rl token: Bootstrapping online rl with vision-language-action models,” *arXiv preprint arXiv:2604.23073*, 2026.
- [33] T. Zhang, C. Yu, S. Su, and Y. Wang, “ReinFlow: Fine-tuning flow matching policy with online reinforcement learning,” *arXiv preprint arXiv:2505.22094*, 2025.

- [34] K. Chen, Z. Liu, T. Zhang *et al.*, “ π_{r1} : Online RL fine-tuning for flow-based vision-language-action models,” *arXiv preprint arXiv:2510.25889*, 2025.
- [35] S. Ding, K. Hu, Z. Zhang, K. Ren, W. Zhang, J. Yu, J. Wang, and Y. Shi, “Diffusion-based reinforcement learning via Q-weighted variational policy optimization,” in *Advances in Neural Information Processing Systems*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.16173>
- [36] H. Ma, T. Chen, K. Wang, N. Li, and B. Dai, “Efficient online reinforcement learning for diffusion policy,” *arXiv preprint arXiv:2502.00361*, 2025. [Online]. Available: <https://arxiv.org/abs/2502.00361>
- [37] Q. Li and S. Levine, “Q-learning with adjoint matching,” in *International Conference on Learning Representations*, 2026. [Online]. Available: <https://openreview.net/forum?id=vd4eNAdtO6>
- [38] Z. Li, S. Tang, and N. Azizan, “Reverse flow matching: A unified framework for online reinforcement learning with diffusion and flow policies,” *arXiv preprint arXiv:2601.08136*, 2026. [Online]. Available: <https://arxiv.org/abs/2601.08136>
- [39] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *International Conference on Learning Representations*, 2016.
- [40] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
- [41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.
- [42] S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration,” in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.
- [43] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, “Stabilizing off-policy q-learning via bootstrapping error reduction,” *Advances in neural information processing systems*, vol. 32, 2019.
- [44] Y. Mao, H. Zhang, C. Chen, Y. Xu, and X. Ji, “Supported trust region optimization for offline reinforcement learning,” in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202, 2023, pp. 23 829–23 851. [Online]. Available: <https://proceedings.mlr.press/v202/mao23c.html>
- [45] A. Kumar, A. Zhou, G. Tucker, and S. Levine, “Conservative q-learning for offline reinforcement learning,” *Advances in neural information processing systems*, vol. 33, pp. 1179–1191, 2020.
- [46] S. Fujimoto and S. S. Gu, “A minimalist approach to offline reinforcement learning,” *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.
- [47] I. Kostrikov, A. Nair, and S. Levine, “Offline reinforcement learning with implicit q-learning,” *arXiv preprint arXiv:2110.06169*, 2021.
- [48] Z. Li, T. Chen, Z.-W. Hong, A. Ajay, and P. Agrawal, “Parallel Q-learning: Scaling off-policy reinforcement learning under massively parallel simulation,” in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202, 2023. [Online]. Available: <https://proceedings.mlr.press/v202/li23f.html>

- [49] M. Nauman, M. Ostaszewski, K. Jankowski, P. Miłoś, and M. Cygan, “Bigger, regularized, optimistic: Scaling for compute and sample efficient continuous control,” in *Advances in Neural Information Processing Systems*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.16158>
- [50] H. Lee, D. Hwang, D. Kim, H. Kim, J. J. Tai, K. Subramanian, P. R. Wurman, J. Choo, P. Stone, and T. Seno, “SimBa: Simplicity bias for scaling up parameters in deep reinforcement learning,” in *International Conference on Learning Representations*, 2025. [Online]. Available: <https://arxiv.org/abs/2410.09754>
- [51] A. Bhatt, D. Palenicek, B. Belousov, M. Argus, A. Amiranashvili, T. Brox, and J. Peters, “CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity,” in *International Conference on Learning Representations*, 2024. [Online]. Available: <https://arxiv.org/abs/1902.05605>
- [52] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017. [Online]. Available: <https://arxiv.org/abs/1707.06347>
- [53] D. P. Kingma and R. Gao, “Understanding diffusion objectives as the elbo with simple data augmentation,” 2023. [Online]. Available: <https://arxiv.org/abs/2303.00848>
- [54] M. Towers, A. Kwiatkowski, J. Balis, G. De Cola, T. Deleu, M. Goulão, K. Andreas, M. Krimmel, A. Kg, R. Perez-Vicente *et al.*, “Gymnasium: A standard interface for reinforcement learning environments,” *Advances in Neural Information Processing Systems*, vol. 38, 2026.
- [55] Amazon FAR, P. Abbeel, J. Chen, R. Duan, A. Escontrela, M. Gandhi, S. Gundry, X. Huang, A. Kanazawa, T. Lewicki, J. Li, K. Liu, C. Rosenthal, Y. Seo, C. Sferrazza, G. Shi, L. Shih, J. Tseng, Z. Wu, L. Yang, B. Yi, and Y. Zhang, “Holosoma,” 2025. [Online]. Available: <https://github.com/amazon-far/holosoma>
- [56] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 449–458.
- [57] H. Jiang, H. Feng, P. Abbeel, J. Jiao, A. Kanazawa, and N. Haghtalab, “DiPOD: Diffusion policy optimization without drifting apart,” *arXiv preprint arXiv:2606.13795*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.13795>
- [58] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-rank adaptation of large language models,” in *International Conference on Learning Representations (ICLR)*, 2022.
- [59] S. Huang, R. F. J. Dossá, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. M. Araújo, “CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms,” *Journal of Machine Learning Research*, vol. 23, no. 274, pp. 1–18, 2022. [Online]. Available: <https://jmlr.org/papers/v23/21-1342.html>
- [60] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal, “Robust motion in-betweening,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 60–1, 2020.
- [61] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, and R. Rombach, “Scaling rectified flow transformers for high-resolution image synthesis,” in *International Conference on Machine Learning (ICML)*, 2024.

- [62] Y. Kirstain, A. Polyak, U. Singer, S. Matiana, J. Penna, and O. Levy, “Pick-a-pic: An open dataset of user preferences for text-to-image generation,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [63] D. McAllister, M. Aittala, T. Karras, J. Hellsten, A. Kanazawa, T. Aila, and S. Laine, “Finite difference flow optimization for RL post-training of text-to-image models,” in *European Conference on Computer Vision (ECCV)*, 2026. [Online]. Available: <https://arxiv.org/abs/2603.12893>

Appendix

A.1 QF3 Implementation Details for Training from Scratch

This section describes the flow actor and critic used when training from scratch, in MuJoCo-v4 and humanoid locomotion. The fine-tuning experiments keep the pretrained policy’s architecture, described in Appendices A.5, A.6, and A.8.

Flow actor and integration. The actor is an MLP that takes the observation, noisy action, and time embedding $[s, x_\tau, \phi(\tau)]$ and outputs the velocity $v_\theta(s, x_\tau, \tau)$. The flow time is encoded by a fixed sinusoidal embedding $\phi(\tau) \in \mathbb{R}^d$ with $d = 64$, half-bank size $H = d/2$, and log-spaced frequencies $\omega_k = \exp(-k \log(10000)/(H - 1))$ for $k = 0, \dots, H - 1$, with $\phi(\tau) = [\sin(\tau\omega_k), \cos(\tau\omega_k)]_k$ stored as a non-trainable buffer. The last layer of the velocity network is initialized to zero, so the initial flow is the identity transport $x_1 = \epsilon$. Rollouts use explicit forward Euler with uniform step $\Delta\tau = 1/N$, $N = 5$: starting from $x^{(0)} = \epsilon \sim \mathcal{N}(0, I)$, we iterate $x^{(k+1)} = x^{(k)} + \Delta\tau v_\theta(s, x^{(k)}, k\Delta\tau)$ and take $x_1 = x^{(N)}$ as the normalized action.

Table A.1 summarizes the actor and critic architectures for the MuJoCo-v4 and humanoid locomotion experiments. The two settings differ in network width, layer normalization, and critic head, but share the flow parameterization, time embedding, activation, and Euler-step count.

Critic parameterization. In MuJoCo-v4, the critic is a standard TD3 twin-Q with target smoothing and delayed actor updates, identical to the CleanRL TD3 baseline. In the locomotion setting (Holosoma), we adopt the FastTD3 distributional critic: each Q-network outputs categorical logits over $K = 101$ atoms uniformly spaced on $[-V_{\max}, V_{\max}]$ with $V_{\max} = 20$, trained with a cross-entropy loss against a projected target distribution. The mean of the predicted distribution is the scalar Q-estimate used by the actor update.

	MuJoCo-v4	Locomotion
Actor MLP hidden width	256	512
Actor MLP hidden layers	2	2
Actor activation	SiLU	SiLU
Actor LayerNorm	no	yes
Time embedding dim.	64	64
Euler steps N	5	5
Critic head	scalar twin-Q (TD3)	distributional twin-Q (FastTD3)
Critic MLP hidden width	256	768
Critic MLP hidden layers	2	2
Critic atoms / support	—	101 over $[-20, 20]$

Table A.1: **QF3 actor and critic architectures.** MuJoCo-v4 uses the CleanRL-based implementation (Appendix A.3); locomotion uses the FastTD3-based implementation for the humanoid and sim-to-real experiments (Appendix A.4).

A.2 Velocity Clipping as an ELBO-Surrogate Trust Region

This section interprets the velocity clip of Sec. 3 as a trust region on an ELBO surrogate of the replay action. For a replay action a_{buf} and Gaussian noise ϵ , define

$$\begin{aligned} x_\tau &= (1 - \tau)\epsilon + \tau a_{\text{buf}}, & u_{\text{buf}} &= a_{\text{buf}} - \epsilon, \\ \delta_\theta &= v_\theta(s, x_\tau, \tau) - u_{\text{buf}}. \end{aligned} \tag{A.1}$$

With the expectation taken over $(s, a_{\text{buf}}) \sim \mathcal{D}$, $\epsilon \sim \mathcal{N}(0, I)$, and $\tau \sim \mathcal{U}(0, 1)$, the dimension-normalized CFM loss is

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E} \left[\frac{1}{d_a} \|v_\theta(s, x_\tau, \tau) - u_{\text{buf}}\|_2^2 \right]. \quad (\text{A.2})$$

The likelihood connection follows from Kingma and Gao [53]. For noised data $z_\lambda = \alpha_\lambda x + \sigma_\lambda \epsilon$, the weighted denoising loss is

$$\mathcal{L}_w(x) := \frac{1}{2} \mathbb{E}_{\lambda \sim p(\lambda), \epsilon} \left[\frac{w(\lambda)}{p(\lambda)} \|\hat{\epsilon}_\theta(z_\lambda; \lambda) - \epsilon\|_2^2 \right], \quad (\text{A.3})$$

where $w(\lambda)$ weights the log-SNR $\lambda = \log(\alpha_\lambda^2 / \sigma_\lambda^2)$ and $p(\lambda)$ is its sampling density under the noise schedule. They show that this loss equals a negative ELBO,

$$\mathcal{L}_w(x) = C - \mathbb{E}_{\lambda \sim \rho_w, \epsilon} [\text{ELBO}_{\theta, \lambda}(z_\lambda)], \quad (\text{A.4})$$

where C is independent of θ and ρ_w is the nonnegative noise-level weighting induced by a monotone weight w . This condition holds for common schedules, including the linear schedule used by QF3. Eq. (A.2) is therefore, up to constants and the normalization d_a , a negative expected ELBO of the noise-augmented data.

The critic term uses the clipped proposal

$$\bar{\delta}_\theta = \text{clip}(\delta_\theta, -\alpha, +\alpha), \quad \bar{v}_\theta = u_{\text{buf}} + \bar{\delta}_\theta. \quad (\text{A.5})$$

Since $\|\bar{\delta}_\theta\|_\infty \leq \alpha$,

$$\frac{1}{d_a} \|\bar{v}_\theta - u_{\text{buf}}\|_2^2 = \frac{1}{d_a} \|\bar{\delta}_\theta\|_2^2 \leq \alpha^2. \quad (\text{A.6})$$

By the CFM–ELBO correspondence, the clipped proposal satisfies the local surrogate bound

$$\mathbb{E} [\text{ELBO}_{\bar{v}_\theta}^{\text{CFM}}(z_\lambda)] \geq C - \alpha^2, \quad (\text{A.7})$$

with the same dimension-normalized scaling as Eq. (A.2). The bound holds for the clipped proposal $\bar{v}_\theta$, not for the policy’s own velocity v_θ , which the flow-matching loss only keeps near it. Because $\bar{v}_\theta$ is built from u_{buf} , which already encodes a_{buf} , Eq. (A.7) bounds a surrogate rather than the likelihood of a_{buf} under the policy. A large CFM residual means that the noised replay action has a low ELBO surrogate under the actor, so clipping restricts value-guided updates to a trust region. The separate CFM loss is still applied to the unclipped velocity v_θ : saturated coordinates receive no Q-gradient but are still pulled back toward u_{buf} .

Action-space clamp. Each actor update draws one (ϵ, τ) pair per replay sample, computes $x_\tau = (1 - \tau)\epsilon + \tau a_{\text{buf}}$, predicts $v_\theta(s, x_\tau, \tau)$, and forms the one-step reconstruction $\hat{x}_1 = x_\tau + (1 - \tau)(u_{\text{buf}} + \bar{\delta}_\theta)$ on which the Q term is evaluated. QF3 clips the residual in *velocity* space, $\bar{\delta}_\theta = \text{clip}(\delta_\theta, -\alpha, \alpha)$, so the endpoint deviation $\hat{x}_1 - a_{\text{buf}} = (1 - \tau)\bar{\delta}_\theta$ shrinks linearly in $1 - \tau$. The ablation in Fig. 3 instead bounds the endpoint deviation directly in *action* space. Enforcing $|\hat{x}_1 - a_{\text{buf}}| \leq \alpha$ uniformly in τ rescales the bound on the policy-buffer gap by $1/(1 - \tau)$, giving $\text{clip}(\delta_\theta, -\frac{\alpha}{1-\tau}, \frac{\alpha}{1-\tau})$. As $\tau \rightarrow 1$ this bound grows without limit, so the same endpoint deviation corresponds to very different velocity magnitudes at different τ . Optimizing the actor under this bound can therefore push v_θ toward sharply different magnitudes across τ , producing the high-curvature flow field in Fig. 3c.

Method	Searched grid	Selected setting (Hopper / Walker2d / Ant / Humanoid)
QSM	$M_q \in \{25, 50, 120\}$	$M_q = 25 / 50 / 50 / 120$
FlowRL	$\lambda \in \{0.01, 0.03, 0.1, 0.3, 1\}$, expectile $\in \{0.7, 0.8, 0.9\}$	$(\lambda, \text{expectile}) = (0.1, 0.9) / (1, 0.8) / (0.3, 0.9) / (0.03, 0.8)$
DIPO	grad.-norm ratio $r \in \{0.03, 0.08, 0.1, 0.3\}$, action-gradient steps $\in \{10, 20, 40\}$	$(r, \text{steps}) = (0.3, 20) / (0.3, 10) / (0.03, 40) / (0.3, 10)$
QF3	$\alpha \in \{0.5, 1, 2\}$, $\lambda_{\text{cfm}} \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5\}$	$(\alpha, \lambda_{\text{cfm}}) = (1, 0.2) / (2, 0.1) / (1, 0.5) / (2, 0.5)$

Table A.2: **MuJoCo-v4 method-specific settings.** Settings not listed are inherited from the shared CleanRL TD3 harness.

A.3 MuJoCo-v4 Settings

All methods in Sec. 4.2 share a CleanRL-based TD3 harness [59], the flow actor and critic of Appendix A.1, the Gymnasium MuJoCo-v4 tasks [54], and the same evaluation protocol. Each run trains for 500k environment steps and is evaluated every 10k steps on 5 deterministic episodes, with 3 seeds per configuration. We tune each method’s hyperparameters per task, selecting the configuration with the highest mean evaluation return over steps 450k–500k, averaged over seeds; Table A.2 lists the grids and the selected settings. Curves show the mean $\pm$ population standard deviation across seeds.

QSM. We implement Q-Score Matching [13] following the authors’ DDPM/JAX code. At noised replay actions, QSM regresses the velocity field onto the scaled critic action-gradient, $v_\theta \rightarrow M_q \nabla_a Q(s, a_\tau)$, without backpropagating through the sampler, and trains a twin critic with TD targets. The score-loss weight M_q sets the magnitude of this target, and its best value differs across tasks (Fig. A.1a).

FlowRL and DIPO. We port FlowRL [11] and DIPO [10] to this setup. FlowRL maximizes Q with a Wasserstein-2 penalty toward high-value replay actions, which it identifies with an expectile value function; we tune the penalty weight λ and the expectile. DIPO improves replay actions by gradient ascent on the critic and refits the actor to them; we tune its action-gradient norm ratio r and the number of action-gradient steps. DIPO uses the flow actor in place of its original diffusion actor.

QF3. We tune QF3’s clip radius α and CFM weight λ_{cfm} over the values in Table A.2. Fig. A.1b shows the return over the last 50k steps for each combination.

A.4 Holosoma Humanoid Locomotion Settings

This section gives the simulator, task, and baseline configuration for the velocity- and motion-tracking experiments in Sec. 5.1.

Simulator and training stack. All locomotion experiments run in the open-source Holosoma humanoid whole-body RL framework [55], which provides (i) a GPU-parallel humanoid simulator, (ii) velocity-tracking and single-motion-tracking tasks for the Unitree G1, and (iii) a tuned FastTD3 [16] baseline. We use Holosoma’s default observation processing, action space (29-DoF joint targets fed to a low-level PD controller), domain randomization, and termination conditions. QF3 replaces the actor inside the same FastTD3 training loop, sharing the replay buffer, distributional twin-Q critic, target-update schedule, and parallel-environment rollouts. The FastTD3 baseline in Fig. 5 uses Holosoma’s tuned configuration unchanged. Velocity-tracking runs train on a single NVIDIA A10G GPU and motion-tracking runs on a single NVIDIA

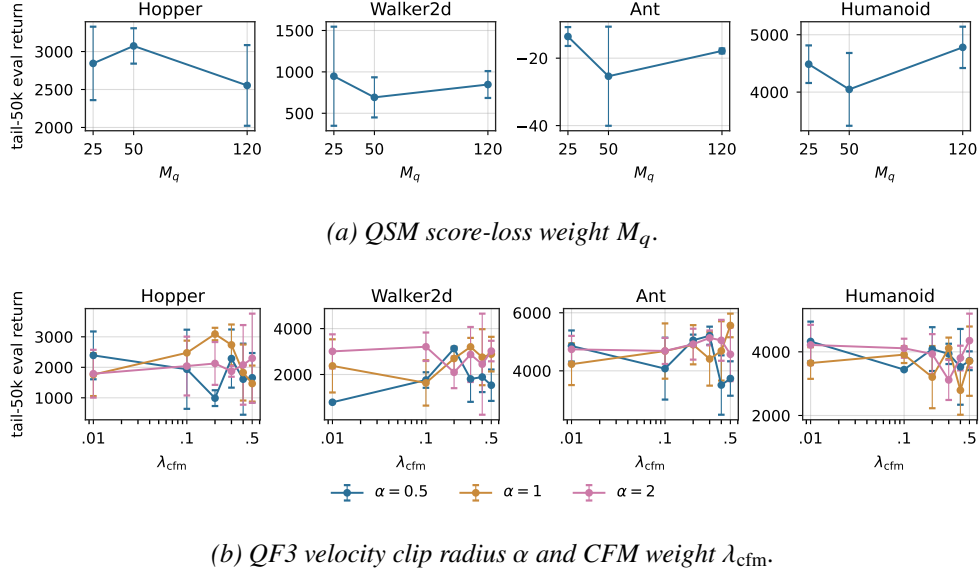


Figure A.1: **MuJoCo-v4 hyperparameter sensitivity.** Tail-50k evaluation return; markers show across-seed means and error bars $\pm 1\sigma$ over 3 seeds.

L40S GPU. Both tasks use the Unitree G1 (29 actuated DoFs) as released in Holosoma. The same URDF, joint limits, and PD gains are used in training and in zero-shot deployment; the policy outputs joint targets at the simulator’s control rate, which the on-robot low-level controller tracks.

Tasks and command distribution. We use Holosoma’s two reference tasks unchanged. Velocity tracking commands a planar base velocity (v_x, v_y, ω_z), resampled per episode from Holosoma’s default command distribution. The y-axis of Fig. 5a is the per-step `raw_lin_vel` reward, Holosoma’s unweighted exponentiated linear-velocity tracking term. Single-motion tracking uses the LAFAN `dance1_subject2` clip [60] as its reference; the y-axis of Fig. 5b is the mean episode length, capped at 500 steps.

QF3 hyperparameters. QF3 and FastTD3 share the actor and critic learning rates, target smoothing ρ , batch size, replay capacity, parallel environment count, gradient steps per environment step, and exploration noise, and on velocity tracking both use Holosoma’s left–right symmetry augmentation. Network widths and the distributional critic are the “Locomotion” column of Table A.1. QF3 uses $\alpha = 2$ and $\lambda_{cfm} = 0.3$ on velocity tracking and $\alpha = 0.5$ and $\lambda_{cfm} = 0.01$ on motion tracking, and integrates the flow with 5 Euler steps.

FPO++ baseline. The FPO++ [9] curves in Fig. 5 are produced with the authors’ open-source code and its released configuration, unchanged. The wall-clock comparison runs all three methods on the same GPU type for each task (A10G for velocity tracking, L40S for motion tracking).

FlowRL baseline. FlowRL [11] uses the same FastTD3 recipe, replacing the actor and its loss with FlowRL’s. Its velocity-tracking runs train on a single NVIDIA L4 GPU rather than an A10G, and its motion-tracking runs on a single NVIDIA L40S, so Fig. 5 omits FlowRL from the velocity wall-clock panel.

A.5 ABC Sim Manipulation Fine-tuning Settings

This section gives the policy, training, baseline, and evaluation configuration for the manipulation experiments in Sec. 5.2. Both tasks use 3 seeds per method.

Base policy. We fine-tune the vision-conditioned ABC-VLA policy [25]. A frozen Gemma+SigLIP VLM (4.3B parameters) conditions an 8-block, 44M-parameter DiT action head. The head integrates a 5-step Euler ODE from Gaussian noise to a 30-step action chunk, of which the first 15 steps are executed. The dashed base-policy level in Fig. 6 pools the step-0 evaluations of all methods.

Trainable parameters. QF3 trains rank-4 LoRA adapters on the attention and MLP projections of the DiT head, 229k trainable parameters in total; the VLM and base head stay frozen. Adapters start at zero, so training begins exactly at the base policy. We omit the q/k/v adapters of the first four blocks.

Actor update. We apply the QF3 objective of Sec. 3 to replay chunks: noise the stored chunk to a random τ , query one velocity, clip the policy-buffer gap to $\pm\alpha$ ($\alpha = 0.5$ on *Bottles*, 0.1 on *Dishrack*), and score the one-step denoised chunk with the critic. Two anchors regularize the update: $\lambda_{\text{cfm}} = 0.01$ toward the replay chunk and $\lambda_{\text{base}} = 1.0$ toward the base head’s velocity with adapters switched off. The unclamped ablation sets $\alpha \rightarrow \infty$ with the same anchors.

Training loop. Each iteration, 16 parallel worlds each collect one episode with the Polyak-averaged target actor ($\tau_{\text{polyak}} = 0.05$), followed by 1600 critic updates (batch 64, learning rate 3×10^{-4}) and 200 actor updates (batch 256, learning rate 4.5×10^{-5}). The replay buffer holds 50k transitions and is pre-filled with base-policy rollouts. No action noise is injected; exploration comes from the flow latent and the lag between the target and online actors.

Residual RL baseline. The flow policy is frozen, and a ResFiT-style MLP (2.8M parameters) reads the cached VLM condition, robot state, and base action and outputs a tanh-bounded correction (scale 0.03 on *Bottles*, 0.01 on *Dishrack*) added to the executed action. It is trained with TD3 on the same critic, replay buffer, simulator, and evaluation protocol, and its zero-initialized output layer makes it start at the base policy.

Tasks and evaluation. An episode succeeds when every target object has been placed at least once, and ends on success or at the timeout. Rewards are per control step and discounted within the action chunk before entering the replay buffer. *Bottles*: five bottles, re-randomized every episode; reward +1 the first time each bottle enters the bin; 1000-step timeout; 50-episode evaluation on fresh layouts after every rollout; runs stopped at 400k environment steps. *Dishrack*: three plates in one fixed layout; reward +1/3 per newly racked plate and +1 on success; 3300-step timeout; 50-episode evaluation every 20k environment steps. Success rate is the fraction of successful episodes. Mean episode length is averaged over successful episodes only (timeouts excluded), so it measures how quickly the policy finishes when it succeeds.

Table A.3 lists the simulation and rollout settings of both tasks.

	<i>Bottles</i>	<i>Dishrack</i>
Control rate	29.4 Hz (500/17)	
Steps executed per policy call	15 (≈ 2 calls/s)	
Parallel worlds, training	16	16
Warm-up worlds $\times$ episodes	80×4	64×2
Eval. worlds $\times$ batches	50×1	10×5
Max. episode length (steps)	1000 (34 s)	3300 (112 s)

Table A.3: **ABC Sim per-task settings.** Each evaluation runs 50 episodes. Warm-up rollouts come from the base policy and pre-fill the replay buffer.

A.6 Robomimic Fine-tuning Settings

This section gives the task, policy, and training configuration for the Robomimic experiments in Sec. 5.2. All methods run in the released OGPO codebase [15] with its Robomimic settings; each task uses 3 seeds per method.

Tasks. *Square* (single-arm peg insertion), *Tool Hang* (single-arm multi-step tool insertion), and *Transport* (bimanual hand-over) use state observations, and policies output end-effector pose deltas and gripper commands at 20 Hz. Every task has a sparse reward of -1 per step until success, and an episode ends on success or at the time limit. Per-task settings are listed in Table A.4.

	<i>Square</i>	<i>Tool Hang</i>	<i>Transport</i>
State / action dim.	14 / 7	14 / 7	28 / 14
Episode limit	400	1000	800
Demonstrations	MH	PH	MH
BC stopped at $\leq 50\%$	yes	yes	yes
Action chunk h	4	8	8
Discount γ	0.99	0.999	0.999
Online steps	1.5M	2M	2M
QF3 clip α	0.12	0.12	0.02

Table A.4: **Robomimic per-task settings.** Task and training settings follow OGPO [15]. MH/PH: multi-/proficient-human demonstrations.

Base policy and warm start. The base policy is an MLP flow policy integrated with 10 flow steps, pretrained with behavior cloning on the demonstrations in Table A.4. All methods start from the same checkpoint. No demonstrations are used during online RL; the replay buffer is seeded with base-policy rollouts before training starts. Under the evaluation protocol below, the base checkpoint succeeds in 53%, 48%, and 48% of episodes on *Square*, *Tool Hang*, and *Transport* (27%, 20%, and 14% with a single deterministic sample).

Shared critic and training loop. All full fine-tuning methods share OGPO’s critic: an ensemble of 10 MLPs (4×512 hidden units; 5×512 on *Transport*), trained by TD regression with the ensemble mean as the target and Polyak averaging ($\tau = 0.05$), with batch size 256. A single (non-vectorized) environment collects training data, and each environment step is followed by one critic and one actor update.

QF3 and QF3+. QF3 replaces OGPO’s actor update with the clipped one-step objective of Sec. 3; its Q term takes the minimum over all 10 critic heads. Before the first actor update, the critic is trained alone on base-policy rollouts for 20k steps on *Square* and *Tool Hang* and 100k steps on *Transport*. QF3+ adds OGPO’s success-buffer term: a conditional flow-matching loss on actions from successful episodes in the replay buffer. Both add the base-policy anchor of Sec. 5.2 with $\lambda_{\text{base}} = 12.5$. QF3+ weights the success-buffer loss by 100, larger than OGPO’s 1.0, because it is balanced against the $-Q$ term, whose magnitude is on the scale of the returns.

OGPO. We run OGPO+CA with the authors’ Robomimic hyperparameters: PPO-style updates over 32 sampled denoising trajectories per state with the critic as the terminal reward, the success-buffer loss (weight 1.0), and conservative advantages, which take the minimum advantage over the critic ensemble when all members agree it is positive, the maximum when all agree it is negative, and zero otherwise.

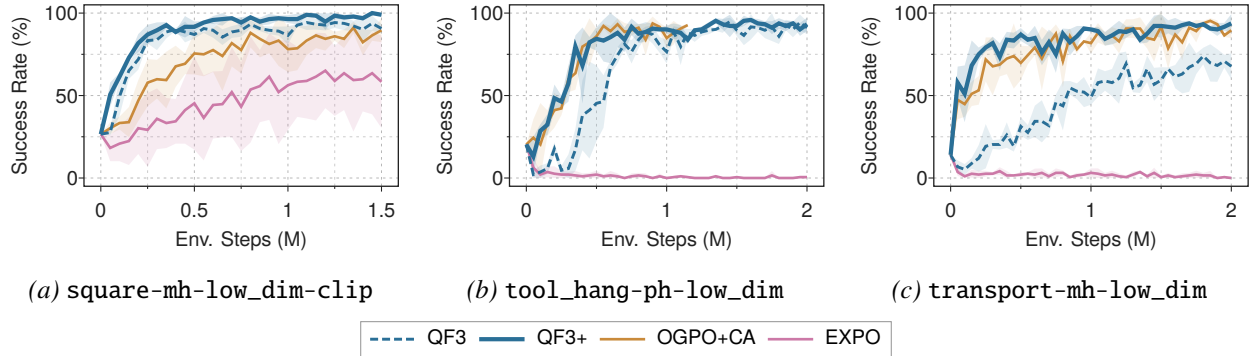


Figure A.2: **Robomimic fine-tuning, deterministic evaluation.** The runs of Fig. 7, evaluated with one ODE sample of the flow policy instead of the policy each method acts with: no SDE noise and no best-of- N selection by the critic. Step 0 is the shared checkpoint (27%, 20%, and 14%). 3 seeds, mean $\pm$ std.

DSRL and EXPO. DSRL [21] and EXPO [14] use the implementations and Robomimic settings of the OGPO codebase and start from the same checkpoint. Neither updates the base policy with RL: DSRL keeps the checkpoint frozen and learns a policy over its input noise, and EXPO learns a separate policy that edits the base policy’s actions.

Evaluation. Every 50k online steps we run 64 evaluation episodes (32 parallel environments, 2 episodes each). QF3, QF3+, and OGPO+CA act with OGPO’s best-of- N sampler, in training rollouts and in evaluation: they draw 8 candidate actions with the SDE sampler and execute the one the critic scores highest. EXPO is evaluated with its on-the-fly policy, which also selects among base and edited samples with the critic, and DSRL samples from its noise policy. Success rate is the fraction of successful episodes. Fig. A.2 reports the same runs evaluated with a single deterministic (ODE) sample of the flow policy, without the critic.

A.7 ABC Sim Speedup Diagnostics

We analyze the gains of Sec. 5.2, and for each task, we run the base policy and a final QF3 checkpoint on the same 50 evaluation seeds.

A.7.1 Load Plates in Dishrack

QF3 raises success from 48% to 72% and shortens the mean episode from 90 to 63 s. About three quarters of this reduction comes from the 15 seeds that the base policy fails and QF3 solves; 21 seeds succeed under both, 3 only under the base policy, and 11 under neither.

QF3 converges on strong behaviors that the base policy already produces. Here, we find QF3’s biggest gains are not from *faster* actions: about four fifths of the per-plate time saved is in manipulation (from the first grasp that moves a plate until it is counted) rather than reach, and on the first plate QF3 even reaches more slowly (4.8 vs. 3.0 s) and carries the plate into the rack no faster (median 5.1–5.2 s for both). Instead, it decreases the number of handoffs. On the first plate, the base policy usually grasps with the left hand, lifts the plate only a few centimeters, regrasps with the right, and takes it back with the left (41 of 50 episodes); QF3 hands it directly from the right hand to the left (49 of 50), a handover the base policy already uses on 4 seeds at nearly the same speed (median 10.6 vs. 9.9 s).

For the dish insertion: on later plates, the base policy often pushes a plate into the rack without seating it in a slot, and then keeps adjusting it, lays it across the wires, or hands it back and retries, sometimes for over a minute (Fig. A.4). QF3’s carry-and-insert is short and consistent (7.3 vs. 12.0 s when successful). In the base model, a plate left more than 20° from upright often slides back out (Fig. A.5b), typically 14 s, or about 27

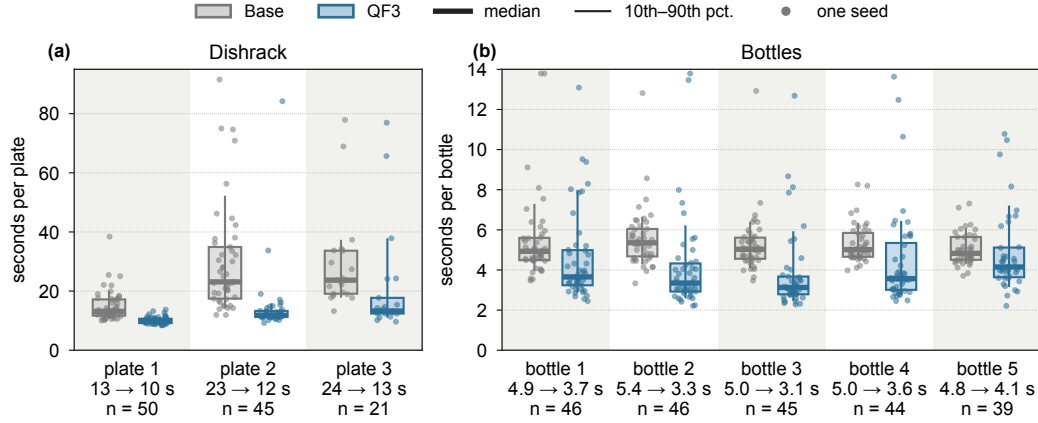


Figure A.3: **Time per object**, from one object being counted to the next, on the seeds where both policies complete that object. Labels give the median for the base policy → QF3 and the number of seeds. *Bottles* uses a separate recorded evaluation and omits the 4 layouts with a bottle counted at the first step; points above 14 s are drawn at the top edge.

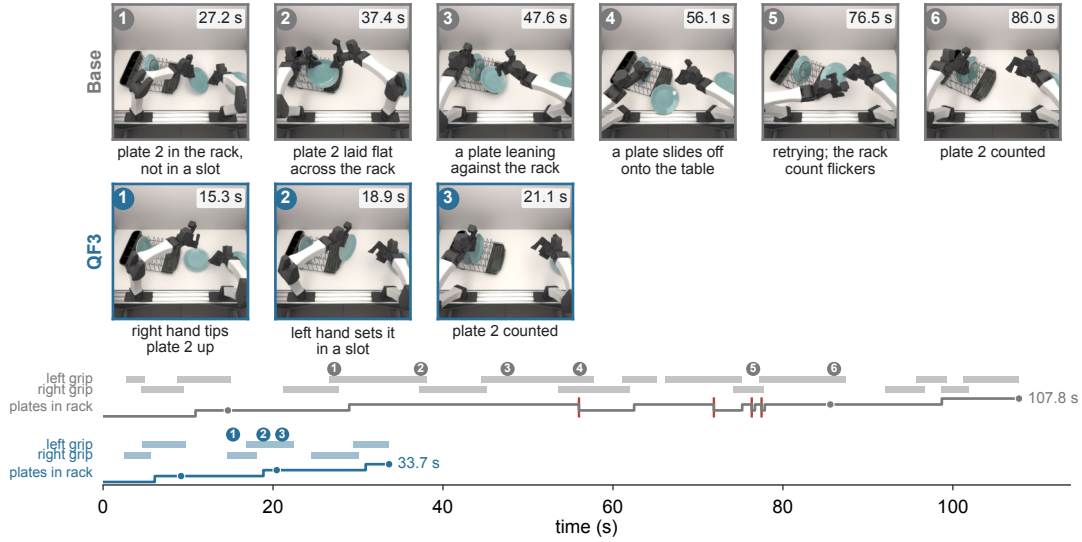


Figure A.4: **One Dishrack seed**. The base policy spends 71 s on the second plate and QF3 11 s. Stills are Blender renders of the recorded states, numbered as on the timeline, which shows gripper closures, plates in the rack (red ticks: a plate leaves), and evaluation counts (dots).

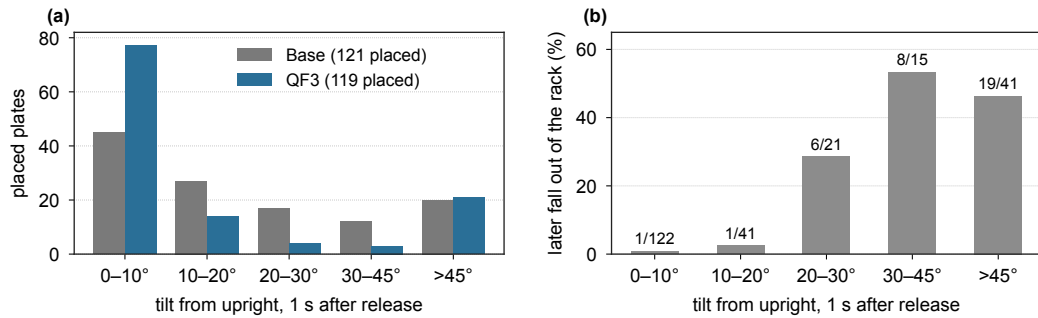


Figure A.5: **Plate tilt on Dishrack**. (a) Tilt from upright 1 s after release, per policy. (b) Share of placed plates in each tilt band that later leave the rack, both policies pooled.

policy calls, after its release and usually with neither gripper within 16 cm (contact with another part of the arm cannot be ruled out). QF3 learns to seat plates nearly upright (Fig. A.5a) and loses 8% of the plates it places, against 21% for the base policy.

A.7.2 Throw Bottles in Bin

QF3 raises success from 86% to 96% and shortens the mean episode from 26.9 to 21.1 s. Unlike on *Dishrack*, about three quarters of this reduction comes from seeds that both policies solve, and the improvement comes from QF3 speeding up episodes. The typical bottle takes 1–2 s less (Fig. A.3b), and is held shorter (1.8 vs. 2.5 s) and grasps sooner.

A.8 Text-to-Image Fine-tuning

We apply the QF3 actor update to text-to-image fine-tuning, as a demonstration beyond robotics.

Setup. We fine-tune Stable Diffusion 3.5 Medium [61] to maximize PickScore [62], keeping the sampler, reward model, and LoRA setup of the FDFO codebase [63]. We treat text-to-image generation as a one-step RL problem: the prompt c is the observation, the final image latent is the action, and the reward model R replaces the learned critic. For a replayed latent a_{buf} , with x_τ , u_{buf} , and δ_θ as in Eq. (A.1) and c in place of s , the actor minimizes

$$-R(D(\hat{x}_1), c) + \lambda_{\text{cfm}} \|\delta_\theta\|^2 + \lambda_{\text{base}} \|v_\theta - v_{\text{base}}\|^2, \quad \hat{x}_1 = a_{\text{buf}} + (1 - \tau) \text{clip}(\delta_\theta, -\alpha, \alpha), \quad (\text{A.8})$$

where D is the VAE decoder and v_{base} is the network with the LoRA adapter disabled. The reward is evaluated at the clipped one-step endpoint, with its gradient flowing through D and PickScore’s image encoder, while the flow-matching term acts on the unclipped residual.

Training. We train a rank-32 LoRA adapter (scale 64) on the transformer’s attention projections with AdamW (learning rate 3×10^{-5} , weight decay 10^{-4} , gradient-norm clip 1.0), following FDFO. Each epoch samples 128 images at 512×512 (40 sampler steps, no classifier-free guidance) for fresh prompts from PickScore’s 25,432 training prompts, adds their latents to a 1024-entry first-in-first-out replay buffer, and takes 16 optimizer steps on 64 replay draws each. We use $\alpha = 0.225$, about half the mean $|\delta_\theta|$ (clipping 63–67% of elements), $\lambda_{\text{cfm}} = 0.01$, and $\lambda_{\text{base}} = 1$, and train for 24 epochs on one NVIDIA H100.

We test on three settings: a replay buffer of size 128, refreshed every epoch; a replay buffer of length 1024, sampled uniformly; and a buffer of length 1024 but where draws favor recent entries, with weight $0.5^{k/128}$ for the k -th newest, such that about half come from the current epoch. We also run the uniform 1024 configuration with the clip removed, so that $\hat{x}_1$ uses the unclipped δ_θ .

Results. We report the results in Fig. A.6, and the results show the mean PickScore of the training samples rises from 19.72 to 22.57; Table A.5 lists the prompts shown in Fig. A.6. Figure A.7 follows the same illustrative seeds (3, 4, 8) at epochs 4, 8, 16, and 24 for the golden-retriever and beaver prompts. All samples use 40 sampler steps, guidance 1.0, and 512×512 generation. These examples illustrate qualitative differences across methods.

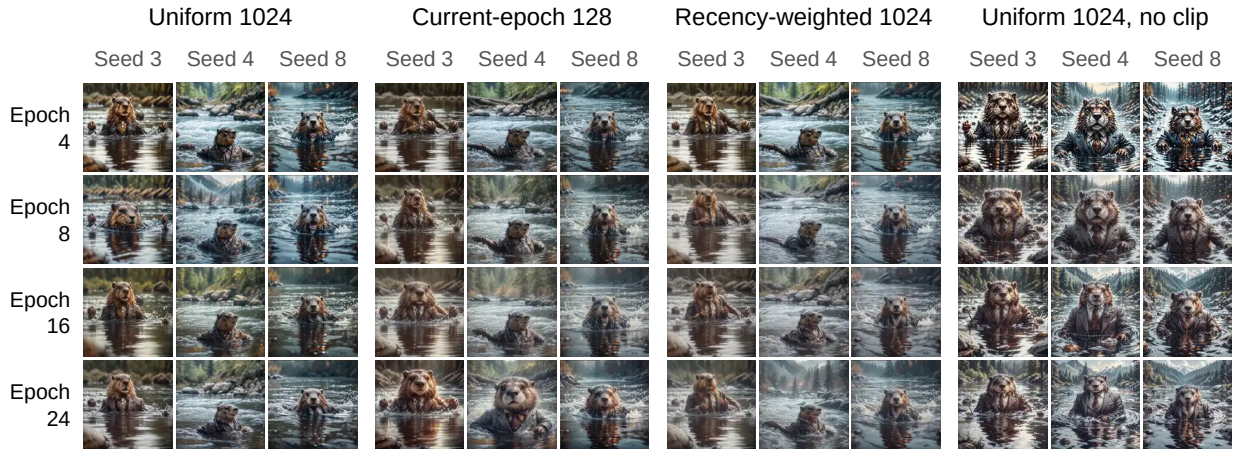
Despite similar training PickScore, the replay configurations appear to follow different qualitative trajectories (Fig. A.7). In the displayed probes, current-epoch replay (128) tends toward more uniform, close-up compositions, reducing variation in subject scale and framing. Uniform replay (1024) retains more varied compositions but exhibits larger fluctuations in color, contrast, and lighting across epochs. Recency-weighted replay (1024) maintains a more consistent visual appearance while retaining variation in framing.



Figure A.6: **Fine-tuning Stable Diffusion 3.5 Medium on PickScore.** (a, b) Six fixed prompts (Table A.5) and seeds before and after fine-tuning. (c) Training reward: mean PickScore of each epoch's training samples.



(a) *Golden retriever.*



(b) *Beaver in a suit.*

Figure A.7: **Qualitative progression under three replay configurations and without the clip.** Columns compare methods; the last is uniform replay (1024) with the clip removed. Rows show epochs 4, 8, 16, and 24. Each method uses the same three seeds.

Position	Prompt
Top left	“a robot doing the tango”
Top middle	“a puffin eating a muffin”
Top right	“golden retriever with a stick in its mouth on a hike to a waterfall”
Bottom left	“a boy kicking a soccer ball”
Bottom middle	“goldfish swimming in a fishtank”
Bottom right	“beaver wearing a suit swimming in a river”

Table A.5: **Prompts in Fig. A.6**, by position in each grid.