SIMPLE FLOW GRADIENT: FINE-TUNING FLOW POLICIES WITH THE DIRECT Q-GRADIENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative policies trained by imitation rarely reach the reliability that deployment demands, and reinforcement learning (RL) offers a way to improve them through feedback from their own interactions. Off-policy methods reuse past experience for sample efficiency, but applying them to a generative policy raises two concerns identified by prior work: backpropagating the critic's gradient through the sampling chain may be unstable, and an update without an anchor to the data may exploit errors in the critic's value estimates. This work examines whether the direct Q-gradient, the most elementary such update, can fine-tune a pretrained generative policy without an anchor. Analyzing direct Q-gradient updates on a one-step flow policy, we find that a single sufficiently large update can erase pretrained competence. This motivates bounding the action space change of each episode's update while allowing continued adaptation beyond the pretrained policy. **Simple Flow Gradient (SFG)** applies this bound by projection to a pretrained flow policy updated with the direct Q-gradient; the bound is its only addition to the actor-critic recipe, which keeps no anchor term, distillation stage, or auxiliary network. Starting from converged behavior cloning policies under matched interaction budgets, SFG reaches a mean task completion score of 0.98 on five manipulation tasks from Robomimic and Franka-Kitchen, and a score of 0.8 more than twice as fast as the fastest baseline. Additional experiments show that SFG also applies to multi-step flow policies.

1 INTRODUCTION

Generative policies, which model actions with diffusion (Chi et al., 2023; Ze et al., 2024; Octo Model Team et al., 2024) or flow matching (Black et al., 2024; Physical Intelligence et al., 2025b; NVIDIA et al., 2025), have been widely adopted for robot manipulation and are typically trained by imitation learning. However expressive, such policies often fall short of the reliability that deployment demands (Luo et al., 2025; Physical Intelligence et al., 2025a). The learned policy may fit the demonstrations imperfectly, and small action errors compound over long horizons, driving the policy into states that the demonstrations do not cover (Ross & Bagnell, 2010; Ross et al., 2011); its performance degrades further under changes in the scene or the task conditions (Xie et al., 2024; Pumacay et al., 2024). Reinforcement learning (RL) offers a way to improve these policies through feedback from their own interactions. How, then, should it be applied to a pretrained generative policy?

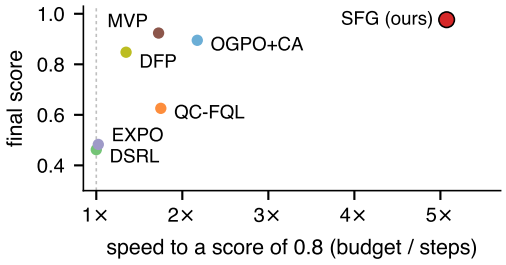


Figure 1: SFG fine-tunes a pretrained flow policy with the direct Q-gradient, no anchor, and one bound on the step of each episode. How many times sooner than the 1.4M budget the task completion score reaches 0.8 (1× if never), against the final score, five-task mean of Table 1.

We focus on off-policy actor–critic methods (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018) rather than on-policy approaches (Ren et al., 2025; Zhang et al., 2025b), as they can

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.

reuse past experience to improve sample efficiency. Applying them to a generative policy raises two concerns: backpropagating the critic's gradient through the sampling chain may be unstable (Patil et al., 2026; Zhang et al., 2026; Doo et al., 2026), and a policy update without an anchor to the data may exploit errors in the critic's value estimates (Wang et al., 2023; Patil et al., 2026; Dong et al., 2026b). These concerns have shaped the update designs of existing methods. Many update the policy indirectly, by steering, editing, or filtering the samples of a base policy (Wagenmaker et al., 2025; Dong et al., 2026a; Li et al., 2025), by regressing onto critic-optimized actions (Mark et al., 2024; Ziakas et al., 2026), by distilling a one-step actor (Park et al., 2025), or by optimizing a surrogate objective over the denoising chain (Patil et al., 2026; Doo et al., 2026; Wang et al., 2026b; Dong et al., 2026b). Even those that backpropagate the critic's gradient directly keep an anchor to the demonstrations or to the pretrained policy (Wang et al., 2023; Zhang et al., 2026). These designs leave aside the simplest alternative: the *direct Q-gradient* without an actor anchor. Can this update effectively fine-tune a pretrained generative policy?

To answer this question, we analyze the direct update on a one-step flow policy, removing the sampling chain and its associated backpropagation difficulties. We find that direct critic optimization can effectively improve pretrained flow policies without an anchor. A single sufficiently large update along the critic's gradient can erase the pretrained competence, which motivates measuring and bounding the action space change of each update. We therefore propose **Simple Flow Gradient (SFG)**, which updates the actor with the direct Q-gradient and bounds the action space change of each episode's update. The bound lets the policy improve progressively without tying it to its pretrained behavior, and it is the only addition to the actor-critic recipe, which keeps no anchor term, distillation stage, or auxiliary network. In our experiments, every method starts from a behavior cloning policy trained to convergence and runs under a matched interaction budget. Across five manipulation tasks from Robomimic and Franka-Kitchen, SFG achieves a mean task completion score of 0.98, compared with 0.92 for the best-performing baseline. It reaches a score of 0.8 more than twice as fast as the fastest baseline (Figure 1). SFG also improves multi-step flow policies with the same update, using full or truncated backpropagation through the sampling chain (Section 5.3, Appendix G).

2 Preliminaries

2.1 Flow Matching and MeanFlow

Diffusion and flow-matching policies represent a policy $\pi(a \mid s)$ implicitly, through a generative process that transforms Gaussian noise into actions. In flow matching (Lipman et al., 2023; Liu et al., 2023), a demonstration pair (s, a) and a noise sample $\epsilon \sim \mathcal{N}(0, I)$ define the linear interpolation $z_t = (1-t)\,a + t\,\epsilon$ for $t \in [0, 1]$, whose conditional velocity is $\epsilon - a$. A state-conditioned velocity field is trained by regression onto this target,

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D},\, \epsilon\sim\mathcal{N}(0,I),\, t\sim\mathcal{U}[0,1]} \big\| v_\theta(z_t, t \mid s) - (\epsilon - a) \big\|^2. \tag{1}$$

At inference, an action is generated by integrating $\mathrm{d}z_t/\mathrm{d}t = v_\theta(z_t, t \mid s)$ from $z_1 = \epsilon$ to $t = 0$, typically with K Euler steps $z_{t-\Delta} = z_t - \Delta\, v_\theta(z_t, t \mid s)$. Backpropagating the critic's gradient through this sampler requires differentiation through the K iterative steps, which prior work has identified as a potential source of instability (Park et al., 2025; Zhang et al., 2026).

MeanFlow (Geng et al., 2025) replaces the velocity by the *average* velocity over an interval, learned by a network $u_\theta(z_t, r, t \mid s)$ from the same data without a teacher or distillation, so that z_t is carried to z_r in one evaluation (Appendix E). With $r = 0$ and $t = 1$ the policy is the one-step generator $a = f_\theta(s, \epsilon) = \epsilon - u_\theta(\epsilon, 0, 1 \mid s)$, $\epsilon \sim \mathcal{N}(0, I)$: a single network evaluation produces the action, and f_θ is an ordinary differentiable function of θ. The output is clipped to the action box $\mathcal{A} = [-1, 1]^d$ inside the decode, and we write f_θ for the clipped map throughout: it is what is executed, what the critic is trained on and evaluated at, and what the probe of Section 3 measures; the clip enters the gradient only through its exact derivative, which is zero on a saturated coordinate.

2.2 Off-Policy Actor–Critic Reinforcement Learning

We consider a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ with state space $\mathcal{S}$, action space $\mathcal{A} = [-1, 1]^d$, transition kernel P, reward function r, discount factor $\gamma \in [0, 1)$, and initial state

distribution ρ_0. The goal is to find a policy that maximizes the expected discounted return $J(\pi) = \mathbb{E}_\pi\big[\sum_{t\geq 0}\gamma^t r(s_t, a_t)\big]$. Following common practice for generative robot policies (Chi et al., 2023; Black et al., 2024) and action-chunked reinforcement learning (Li et al., 2025), the policy predicts a chunk of H consecutive actions every H environment steps and executes it open loop. We write a chunk as $a \in \mathcal{A}^H$, call it an action, and let $Q^\pi(s_t, a_t)$ be the expected discounted return of executing a_t from s_t and following π thereafter.

Off-policy actor–critic methods (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018) approximate Q^π with a critic Q_ϕ trained on transitions from a replay buffer $\mathcal{D}$ by minimizing the temporal-difference error

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s_t, a_t, s_{t+H})\sim\mathcal{D}}\Big[\big(Q_\phi(s_t, a_t) - r_t^H - \gamma^H\, Q_{\bar{\phi}}(s_{t+H}, a')\big)^2\Big], \qquad r_t^H = \textstyle\sum_{k=0}^{H-1}\gamma^k r_{t+k}, \tag{2}$$

where $\bar{\phi}$ denotes slowly updated target parameters and $a' \sim \pi(\cdot \mid s_{t+H})$. When the policy is a differentiable generator $a = f_\theta(s, z)$ of the state and a noise variable $z \sim p(z)$ (Section 2.1), it can be improved by ascending the critic along its action gradient, as in the deterministic policy gradient and its reparameterized variants (Silver et al., 2014; Haarnoja et al., 2018). The actor minimizes

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{s\sim\mathcal{D},\, z\sim p(z)}\Big[\min_{i\in\mathcal{M}} Q_{\phi_i}\big(s, f_\theta(s, z)\big)\Big] + \lambda\,\mathcal{R}(\theta; \mathcal{D}), \tag{3}$$

where the critic is an ensemble of N networks and $\mathcal{M} \subset \{1, \ldots, N\}$ a random subset of M of them, drawn afresh for every update; the same subsampled minimum over the target networks replaces $Q_{\bar{\phi}}$ in equation 2. The subsampled minimum in the target is the aggregation of REDQ (Chen et al., 2021), which RLPD adopts. REDQ and RLPD then update the actor against the ensemble mean; we use the subsampled minimum in the actor loss as well, a choice of this paper, and it is the only form of pessimism in the update. The term $\mathcal{R}$ pulls the policy toward the data or toward the pretrained policy, as a generative behavior cloning loss on demonstrations, a diffusion loss or the flow matching loss equation 1 (Wang et al., 2023; Nguyen & Yoo, 2026; Wang et al., 2026a), a distillation loss (Park et al., 2025), a proximity penalty (Zhang et al., 2026), a loss on successful online episodes (Patil et al., 2026), or a hard radius around the pretrained policy (Ziakas et al., 2026; Dong et al., 2026b); we call any such term an *anchor* and an update with $\lambda = 0$ *anchor-free*.

2.3 Online RL with Offline Data

When demonstrations are available, RLPD (Ball et al., 2023) shows that an off-policy actor–critic can exploit them without offline pretraining and without any constraint on the policy: each batch is drawn half from the demonstrations and half from online transitions, and the critic is a layer-normalized ensemble with a pessimistic target (Chen et al., 2021). RLPD trains a Gaussian policy from scratch on top of this critic. We adopt its replay and critic unchanged and replace the actor with a pretrained generative policy. The demonstrations stay in the replay buffer, and the actor draws its states from the same mixed batches as the critic, so demonstration states do enter the actor loss; what is absent is any term that pulls the generated actions toward the demonstrated ones or toward the pretrained policy.

3 Where the Direct Update Fails

3.1 Problem Setting

For a cleaner analysis, we use a one-step generator (Section 2.1) to remove the iterative sampling chain, avoiding the BPTT difficulties highlighted in prior work (Zhang et al., 2026; Doo et al., 2026). What remains to test is the anchor, which one-step generators have kept as well (Nguyen & Yoo, 2026; Wang et al., 2026a). Multi-step sampling and alternative backpropagation schemes are evaluated in Section 5.3.

All runs in this section start from one generator θ_0 on `tool_hang`, a state-based manipulation task from Robomimic (Mandlekar et al., 2021), pretrained to convergence with the MeanFlow objective; unless otherwise stated, they use the critic and replay scheme of Section 2.3.

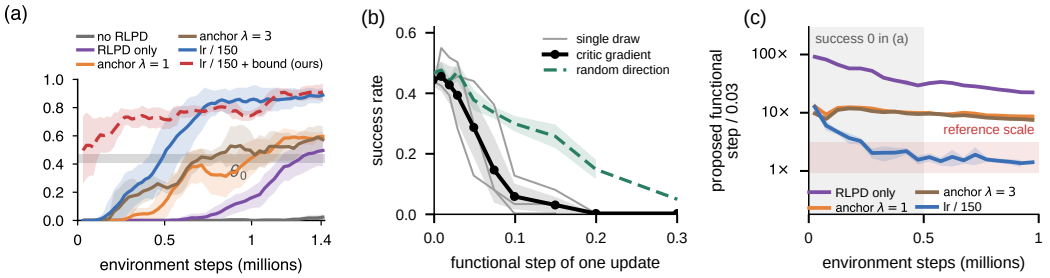


Figure 2: `tool_hang`, every run from one pretrained θ_0. (a) Success of direct-update variants, with SFG (dashed) as a method reference. The smaller-learning-rate runs use $\tau = 0.005$ without the bound and $\tau = 0.05$ for SFG; Table 2 isolates the bound at matched τ. The gray band is θ_0. (b) Success of θ_0 moved along frozen gradients of a good critic, single draws (gray) and their mean (black), and along random parameter directions at the same functional step (dashed). (c) Proposed functional steps before projection for the settings of (a), normalized by 0.03, a reference scale from (b) (red band); the gray interval is the phase of zero success of the unanchored runs.

3.2 The Direct Update Destroys What Pretraining Built

We fine-tune with the actor loss equation 3 at an actor learning rate of 4.5×10^{-5} from a prior that succeeds about half of the time, and remove one component at a time (Figure 2a). With RLPD and a behavior cloning anchor toward the demonstrated actions (Appendix A), success drops to zero within the first evaluations, remains there for a fifth to a third of the budget, and then recovers to 50 to 60%. Without the anchor, the zero-success phase stretches to roughly half of the budget, and the endpoint changes by less than the seed-to-seed spread (Appendix A). Without RLPD as well, so that the demonstrations are kept out of every batch, the critic's and the actor's, success stays at zero for the whole budget.

Finding 1. With demonstration replay, the direct update loses the pretrained competence before relearning, both with and without an actor anchor, and ends near the level it started from; the anchor shortens the zero-success phase but does not remove it.

3.3 A Single Step Is Already Too Large

How large a step along the critic gradient can θ_0 tolerate? To examine the effect of step size with a fixed critic, we take a well-trained critic (Appendix A), freeze it, and compute the direct update once at θ_0. We then move θ_0 along that gradient by a chosen distance and evaluate, with no training in between (Figure 2b). A step of a hundredth of a percent of the parameter norm already halves its success rate, although the successful run ends hundreds of times further from θ_0. This motivates measuring a policy update by its effect on the generated actions: for a change of the actor from θ to θ' we define the *functional step*

$$d(\theta, \theta') = \Big(\frac{1}{|\mathcal{B}|} \sum_{(s,\epsilon)\in\mathcal{B}} \big\| f_{\theta'}(s,\epsilon) - f_{\theta}(s,\epsilon) \big\|^2 \Big)^{1/2}, \tag{4}$$

the root-mean-square change of the generated action chunk over a fixed probe bank $\mathcal{B}$ of states and noise draws, sampled once from the demonstrations and held fixed for the whole run (Appendix A). Small functional steps preserve the pretrained policy's performance, whereas larger steps lead to a sharp drop in success (Figure 2b). At matched functional step sizes, random directions cause a smaller loss of success than critic-gradient directions, although performance declines along both as the step grows. A random direction needs about a hundred times the parameter displacement of the gradient to move the actions by the same functional step. In the runs of Figure 2a, the optimizer proposes much larger functional steps, particularly in the first episodes, when the pretrained competence is lost (Figure 2c). The anchor reduces these steps severalfold, but the initial collapse persists.

Finding 2. Small functional steps preserve pretrained performance in the probe, while the updates of direct fine-tuning induce much larger changes in the generated actions.

A first remedy is a smaller learning rate. With a 150-fold smaller learning rate, success still collapses at the start, for the first 150 to 250 thousand steps, but then recovers faster and further than any anchored run, to 85% at one million steps against 60% (Figure 2a). Even at a fixed learning rate, the proposed functional steps vary substantially over training: at the small rate they are eight times larger early in training than later (Figure 2c). This variation motivates bounding the functional step directly, as developed in Section 4.

4 Method: Bounding the Functional Step

4.1 Functional Step Projection

Motivated by Section 3, we impose a bound δ on the functional step equation 4 of each episode's update and enforce it through *functional step projection*. Two copies of the actor take part. Within each episode, the *proposal* actor $\tilde{\theta}$ collects the data and receives the gradient updates, while the *committed* actor θ stays fixed as the reference. At the episode boundary, the accumulated update $\Delta\theta = \tilde{\theta} - \theta$ is rescaled to satisfy the functional step bound: the committed actor becomes

$$\theta^+ = \theta + \alpha\,\Delta\theta, \qquad d(\theta, \theta^+) \le \delta, \tag{5}$$

with $\alpha = 1$ when the proposal already satisfies the bound, and otherwise an $\alpha \in (0, 1)$ satisfying $d(\theta, \theta + \alpha\,\Delta\theta) = \delta$, found by a scalar root search on the probe bank. In practice, the projection is solved approximately, to a numerical tolerance (Appendix F). The bound limits the change of the committed actor from one episode to the next on the probe bank; the proposal that collects the episode is not bounded within it. The reference advances with the committed actor, allowing continued adaptation beyond the pretrained policy. In action space the bound shortens the path the policy travels by more than an order of magnitude without keeping it near θ_0, which is what an anchor does (Appendix B). The proposal is then reset to θ^+, with the optimizer state kept, and the next episode begins. We use an actor learning rate of 3×10^{-7} and $\delta = 0.01$ on every task.

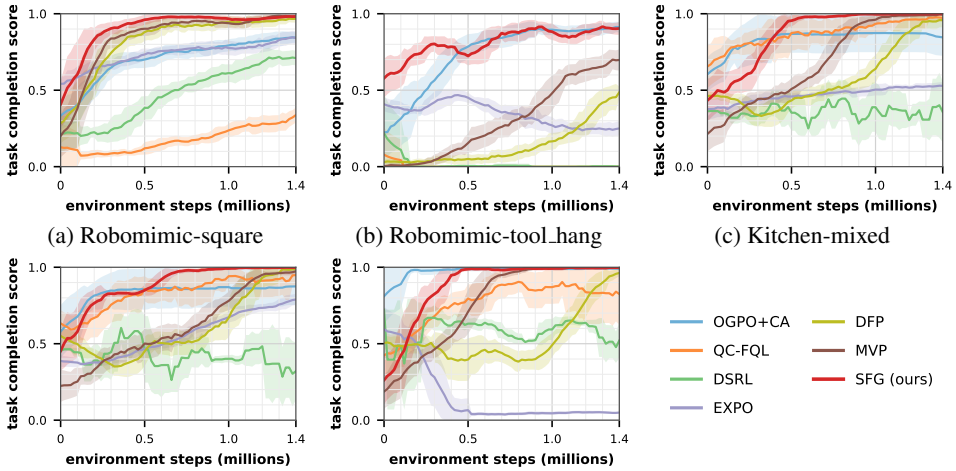
4.2 The Full Algorithm

Simple Flow Gradient (SFG) adopts the critic and mixed replay of RLPD (Section 2.3), retaining access to demonstration experience when online success is scarce (Vecerik et al., 2017; Nair et al., 2018). The actor is a pretrained flow policy with its observation encoder frozen, trained on the direct update, equation 3 with $\lambda = 0$. Its loss is normalized by the batch mean of $|Q|$ (Appendix F), and its updates are applied through the functional step projection of Section 4.1. The generator may equally be a K-step sampler: the functional step is then measured on its output, and nothing else in Algorithm 1 changes. Algorithm 1 gives the full procedure and Appendix F the implementation details and hyperparameters.

Algorithm 1 Simple Flow Gradient (SFG): anchor-free fine-tuning with functional step projection

Require: generator θ_0; demonstrations $\mathcal{D}_{\text{demo}}$; random critic ϕ; bound δ; probe bank $\mathcal{B}$

1: $\theta, \tilde{\theta} \leftarrow \theta_0$; $\bar{\phi} \leftarrow \phi$; $\mathcal{D}_{\text{on}} \leftarrow W$ steps with θ_0, then W critic updates (lines 7–8), $\tilde{\theta}$ fixed ▷ warm-up
2: **for** each episode **do**
3: $\quad\tilde{\theta} \leftarrow \theta$ ▷ the proposal starts from the committed actor
4: $\quad$**for** each action chunk, every H environment steps **do**
5: $\quad\quad a \leftarrow f_{\tilde{\theta}}(s, \epsilon)$, $\epsilon \sim \mathcal{N}(0, I)$ ▷ a new chunk
6: $\quad\quad$**for** each of the H actions of a, after executing it **do** ▷ one critic and one actor update per step
7: $\quad\quad\quad$sample H-step segments (s, a, r^H, s_{+H}), half $\mathcal{D}_{\text{demo}}$ and half $\mathcal{D}_{\text{on}}$; draw head subsets $\mathcal{M}, \mathcal{M}'$
8: $\quad\quad\quad$update ϕ on equation 2 with target $\min_{i\in\mathcal{M}} Q_{\bar{\phi}_i}$, $a' = f_{\tilde{\theta}}(s_{+H}, \epsilon')$; $\quad\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$
9: $\quad\quad\quad$update $\tilde{\theta}$ by one optimizer step on $-\mathbb{E}_{s,\epsilon}\big[\min_{i\in\mathcal{M}'} Q_{\phi_i}\big(s, f_{\tilde{\theta}}(s,\epsilon)\big)\big] \big/ \operatorname{sg}\mathbb{E}|Q|$
10: $\quad\quad$**end for**
11: $\quad$**end for**
12: $\quad$add the episode's transitions to $\mathcal{D}_{\text{on}}$
13: $\quad\theta \leftarrow \theta + \alpha\,(\tilde{\theta} - \theta)$, with $\alpha = 1$ if $d(\theta, \tilde{\theta}) \le \delta$, else the root of $d\big(\theta, \theta + \alpha(\tilde{\theta} - \theta)\big) = \delta$
14: **end for**



(a) Robomimic-square (b) Robomimic-tool_hang (c) Kitchen-mixed

(d) Kitchen-partial (e) Kitchen-complete

Figure 3: Task completion score over the budget on the five tasks of Table 1, under the protocol of Section 5.1. Ours is the one-step version of SFG with $\delta = 0.01$, starting from the pretrained θ_0 at step 0.

5 Experiments

We evaluate SFG's final performance and sample efficiency against existing fine-tuning methods, then examine the roles of its components and sampler.

5.1 Setup

Tasks. We use six sparse-reward manipulation tasks. From Robomimic (Mandlekar et al., 2021): `square`, a medium-horizon insertion; `tool_hang`, a long-horizon multi-step insertion that demands precision; and `transport`, a bimanual long-horizon transfer; `square` and `transport` use the multi-human demonstrations, `tool_hang` the proficient-human ones. From Franka-Kitchen (Gupta et al., 2019): `complete`, whose demonstrations perform all four subtasks in order; `mixed`, where the order varies; and `partial`, where many demonstrations stop early. Reward is one on success. We report a *task completion score*: the success rate on the Robomimic tasks and the fraction of the four subtasks completed on the kitchen tasks; means over tasks average the two.

Baselines. We compare against six generative-policy fine-tuning methods: OGPO+CA (Patil et al., 2026), which fine-tunes the full policy; QC-FQL (Li et al., 2025), which distills a one-step actor; DSRL (Wagenmaker et al., 2025) and EXPO (Dong et al., 2026a), which steer or edit a frozen policy; and MVP (Zhan et al., 2026) and DFP (Koo et al., 2026), which train one-step generators with anchors to the data. Each baseline uses the authors' implementation and published defaults under the same interaction budget. Table 6 in Appendix H summarizes their update mechanisms and actor constraints.

Ours. Unless otherwise stated, we use the one-step version of SFG (Algorithm 1): the MeanFlow generator pretrained to convergence, $\delta = 0.01$, and an actor learning rate of 3×10^{-7} on every task. The multi-step versions (Appendix G) appear on `transport` (Section 5.2) and in the sampler ablation (Section 5.3).

Protocol. Every method gets 1.4M online environment steps and starts from its own behavior cloning policy trained to convergence, whose success is the point at step zero. Policies are evaluated every 20k steps on 64 episodes; Every reported number is the mean over four seeds, for the baselines as for SFG.

Method	Policy frozen	Anchor	Completion score	Steps to 0.8 (M)
DSRL	yes	no	0.46	1.40
EXPO	yes	yes	0.48	1.37
QC-FQL	no	yes	0.63	0.80
DFP	no	yes	0.85	1.04
MVP	no	yes	0.92	0.81
OGPO+CA	no	yes	0.90	0.64
SFG (ours)	no	**no**	**0.98**	**0.28**

Table 1: Main results of one-step SFG, five-task means (per task: Table 7; `transport` is reported separately in Figure 4): task completion score at the end of the budget and environment steps (M) to a score of 0.8 (1.4M if never), with whether each method freezes the pretrained policy and keeps an anchor (Appendix H).

5.2 Main Results

SFG matches or exceeds the best baseline in final task completion score on all five tasks and reaches a score of 0.8 earliest on average (Table 1, Figure 3). Its sample-efficiency gains are largest on `square` and `tool_hang`, while OGPO+CA reaches the threshold earlier on `kitchen-complete`. This compares complete recipes; the differences are not attributed to the update rule or the bound alone. The curves show what the endpoints do not: on `tool_hang`, the three baselines that start from a pretrained policy as good as ours (DSRL, QC-FQL, EXPO) first fall to zero, whereas the smoothed success of SFG never drops below its starting point; OGPO+CA, DFP, and MVP start from much weaker pretrained policies and climb.

Transport. On `transport` the evaluated one-step methods achieve low final success: the one-step version of SFG, whose generator starts at 5%, stays at zero at every evaluation, and QC-FQL, DFP, and MVP end below 15%, as does DSRL (Figure 4); only OGPO+CA and EXPO achieve substantially higher success. We therefore run the multi-step version of SFG from a pretrained flow-matching policy decoded with three Euler steps: with the same direct Q-gradient and functional step bound it climbs from 54 to 81%, below OGPO+CA's 87% (other configurations: Appendix D). The two versions use different pretrained policies, so this comparison demonstrates applicability to multi-step sampling without isolating the effect of sampler depth.

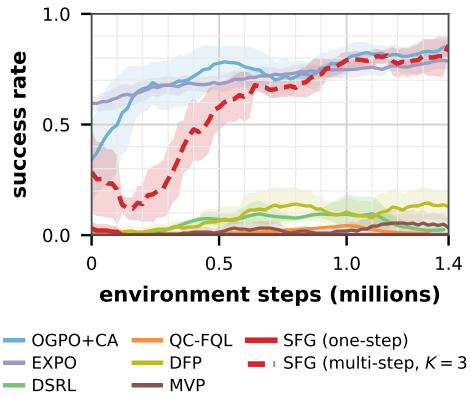


Figure 4: Robomimic-`transport`: the one-step and the multi-step ($K = 3$) versions of SFG against the baselines.

5.3 Ablations

Table 2 removes or varies one component of SFG at a time on `tool_hang`, where the pretrained policy succeeds half the time, and `kitchen-partial`, where it never completes all four subtasks. Cells are the success rate over the first 0.3M steps and over the last 200k steps as in Table 1, each the mean over four seeds, at the learning rate of 3×10^{-7} except in the learning-rate rows. The comparison with and without the bound fixes $\tau = 0.05$; curves and details are in Appendix C.

The functional step bound. Without it, the direct update loses the pretrained competence on `tool_hang`, staying at zero for 0.4M steps before recovering to the same endpoint; with $\delta = 0.01$ the run never falls below θ_0. On `kitchen-partial`, where the pretrained policy has zero full-task success, the run without the bound improves earlier, while both variants reach similar final performance; the clearest benefit of the bound is therefore the preservation of early performance on `tool_hang`. Changing the bound alters the trade-off between early preservation and subsequent improvement; we use $\delta = 0.01$ as the default.

Learning rate. Among the tested learning rates, none without the bound preserves the early success of the run with the bound on `tool_hang`, and the rates small enough to help there slow learning or

		tool_hang		kitchen-partial	
		first 0.3M	final	first 0.3M	final
pretrained θ_0		0.50		0.00	
SFG	$\delta = 0.01$	0.65	0.90	0.21	0.99
bound	none	0.00	0.90	0.36	0.97
	$\delta = 0.005$	0.69	0.83	0.07	0.85
	$\delta = 0.05$	0.37	0.83	0.66	0.96
no bound, learning rate	10^{-7}	0.22	0.84	0.04	0.97
	3×10^{-8}	0.56	0.88	0.11	0.66
	10^{-8}	0.43	0.85	0.00	0.01
data	no demonstration replay	0.48	0.65	0.00	0.00

Table 2: Ablations of SFG on `tool_hang` and `kitchen-partial`. `kitchen-partial` success requires completing all four subtasks.

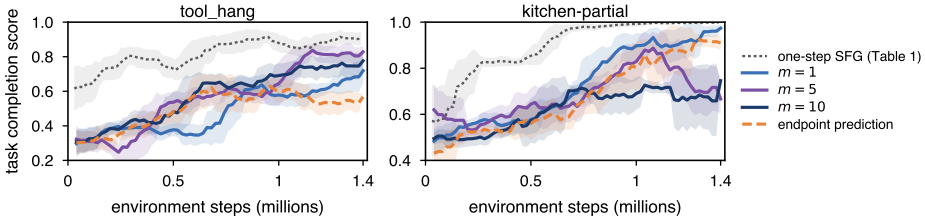


Figure 5: SFG on a ten-step flow policy, by the rule that carries the critic's gradient through the sampler: backpropagation through the last m steps (Algorithm 2) or endpoint prediction (Algorithm 3); the one-step reference is the SFG run of Table 1, a separately pretrained MeanFlow policy. `kitchen-partial` reports the mean fraction of subtasks completed, as in Table 1.

lower the final performance on `kitchen-partial`: a fixed rate cannot adapt the step to the task and the stage of training (Section 3.3).

Demonstration replay. Removing the demonstration half of every batch leaves `tool_hang` at 0.65 and `kitchen-partial` at zero, and removing the bound as well gives zero on both: the bound limits how far an update moves the policy, and demonstration replay supplies successful experience for value learning while online successes are scarce.

Beyond state inputs. On the point cloud `bucket` task of DexArt (Bao et al., 2023), with the 255M-parameter architecture of DP3 (Ze et al., 2024) and a MeanFlow head, the pretrained policy succeeds in 47% of episodes and its PPO demonstrator in 65% (Figure 6; dots are 100-episode holdouts, the band the demonstrator's 95% interval). With the bound, success never collapses and reaches the demonstrator's level within about 400 episodes; without it, success first falls close to zero and regains the pretrained level only after about 450 episodes, even at a hundredfold smaller learning rate.

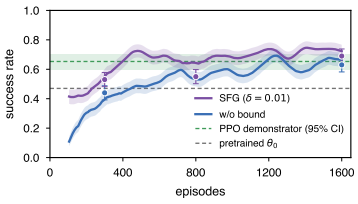


Figure 6: DexArt-`bucket` from point clouds.

Multi-step samplers. All variants execute the same ten-step sampler; the backpropagation depth m determines how many of its final steps receive gradients, and endpoint prediction instead evaluates the critic on actions predicted from intermediate states, without backpropagating through the sampling chain. SFG also improves ten-step flow policies with full or truncated backpropagation (Figure 5). Part of the `tool_hang` runs were throttled by the projection solver (Appendix G), so we do not rank the depths. These runs show that the update trains through a multi-step sampler, not that it matches the stability or the final performance of the one-step version; Appendix G reports all six tasks (Table 5).

6 Related Work

RL fine-tuning of generative policies. On-policy methods apply policy gradients to the denoising chain as an inner MDP (Ren et al., 2025; Zhang et al., 2025b) or through a flow-matching surrogate for the likelihood ratio (McAllister et al., 2026), at a high sample cost (Patil et al., 2026). Most off-policy methods avoid differentiating the critic through the sampler: they learn a correction around the fixed pretrained policy (Wagenmaker et al., 2025; Dong et al., 2026a; Li et al., 2025), extract a policy from the critic by reweighting, distillation, or a one-step actor (Hansen-Estruch et al., 2023; Mark et al., 2024; Park et al., 2025), or replace the critic gradient by a surrogate over the chain (Patil et al., 2026; Doo et al., 2026; Wang et al., 2026b); those that do differentiate through it keep a behavior cloning or proximity term (Wang et al., 2023; Zhang et al., 2026). One-step MeanFlow generators remove the chain but keep a term on the data, flow matching in OFQL, MeanFlowQL, and MVP (Nguyen & Yoo, 2026; Wang et al., 2026a; Zhan et al., 2026) and behavior cloning in DFP (Koo et al., 2026). Our actor is the pretrained generator itself, updated by the critic gradient alone; Appendix H tabulates what each of these methods trains and what holds it.

Why fine-tuning fails, and the anchor. Prior work attributes failures to backpropagation through the sampler (Park et al., 2025; Dong et al., 2026a; Zhang et al., 2026), to critic errors that an expressive actor over-optimizes (Patil et al., 2026; Zhang et al., 2025a; Dong et al., 2026b), to the offline-to-online transition (Nakamoto et al., 2023; Zhou et al., 2025; Wołczyk et al., 2024), to exploration under sparse reward (Zhang et al., 2025b; 2026), and to saturation of bounded actions (Hausknecht & Stone, 2016; Wang et al., 2026a). The common remedy pulls the policy toward the data: a behavior cloning loss inherited from offline RL (Wang et al., 2023; Hansen-Estruch et al., 2023; Park et al., 2025), a proximity penalty (Zhang et al., 2026), or a loss on successful online episodes (Patil et al., 2026). For Gaussian or deterministic policies, TD3-C and PROTO replace this term by a bound on the step from the previous policy (Luo et al., 2023; Li et al., 2023). We study whether controlling changes in generated actions can likewise support fine-tuning of pretrained flow policies without an actor anchor. Existing constraints for pretrained flow policies penalize deviation from the pretrained policy (Ziakas et al., 2026; Dong et al., 2026b; Sun & Song, 2026), whereas trust regions centered at the previous iterate bound each step (Schulman et al., 2015; Sun et al., 2025); SFG measures the change in generated actions relative to the previous committed actor, so the reference advances during fine-tuning (Appendix I).

7 Conclusion and Limitations

We asked whether the direct Q-gradient can fine-tune a pretrained flow policy without an anchor. On a one-step generator, a single sufficiently large update along the critic's gradient erases the pretrained competence; bounding that change once per episode, with no anchor, no distillation, and no extra network, improves the early evaluations of the committed policy in the matched comparison on `tool_hang`. The resulting recipe, SFG, starts from behavior cloning policies trained to convergence, reaches 80% success sooner and ends higher than the compared methods on five manipulation tasks, and carries over to multi-step flow policies and to point-cloud inputs. In these settings, an actor anchor toward the demonstrations or the pretrained policy was not needed for effective learning; what the bound adds is the preservation of the pretrained competence through the first updates.

Four limitations remain. The ablations cover two tasks and do not apportion the five-task gains among the components of the full configuration; the critic aggregation and the loss normalization are listed as settings (Appendix F) rather than ablated. All experiments are in simulation, with state observations except for one point-cloud task; nothing in the recipe is specific to simulation, and a real-robot study is the natural next step. The multi-step version of SFG trains, but its runs neither match the final performance of the one-step version nor establish the same stability, and part of the `tool_hang` runs were affected by the projection solver (Appendix G); they show applicability only. Finally, the bound δ is a fixed value chosen once and reused across tasks, and choosing it took some tuning (Table 2); adapting it automatically during fine-tuning is a direction we plan to explore.

AI USE STATEMENT

We used generative AI tools (Claude, Anthropic; GPT, OpenAI) as writing and engineering assistants under the authors' direction. For writing, the tools were used to edit, condense, and polish text drafted by the authors, to draft passages from the authors' notes and experimental records, and to check the consistency of terminology, notation, and cross-references; every passage was reviewed and revised by the authors, who are responsible for all claims. For engineering, the tools were used to write and refactor the scripts that generate the figures and tables from logged experimental data and to organize experiment records; the generated code was read, tested against the underlying logs, and corrected by the authors. We did not use generative AI tools to generate the research idea, the method, or the analysis of Section 3, to design or run experiments, or to produce, alter, or select experimental results; all numbers in the paper come from the authors' training and evaluation runs. Literature search and the citations were carried out and verified by the authors; where an AI tool summarized a paper, the authors checked the summary against the source. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, and artifacts.

ETHICS STATEMENT

This work studies reinforcement learning fine-tuning of robot manipulation policies in simulation. It involves no human subjects, no personal data, and no real-robot deployment; all experiments use publicly available benchmarks (Robomimic, Franka-Kitchen, and DexArt) and their released demonstration datasets under their respective licenses. We do not foresee ethical risks beyond those common to robot learning research; as with any policy-improvement method, deployment on physical systems would require the usual safety measures, which this paper does not address.

REPRODUCIBILITY STATEMENT

The method is specified in Algorithm 1, with the functional step definition in Section 3.3 and every implementation detail, including the projection solver, the critic, the warm-up, and all hyperparameters, in Appendix F. The analysis of Section 3 follows the protocol of Appendix A, which lists the pretrained policy, the probe bank, the run counts, and the evaluation procedure; the multi-step variants are given as Algorithms 2 and 3 with their setup in Appendix G. Baselines run in their authors' code with published defaults, and Appendix H records what each trains and constrains together with the per-task results behind Table 1; the ablation curves and details are in Appendix C. All tasks and demonstration datasets are public. Our code, with the configuration files of every run, is included as supplementary material; pretrained checkpoints and logged evaluation curves will be released upon publication.

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 (ICLR)*, 2018.

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

Chen Bao, Helin Xu, Yuzhe Qin, and Xiaolong Wang. DexArt: Benchmarking generalizable dexterous manipulation with articulated objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double Q-learning: Learning fast without a model. In *International Conference on Learning Representations (ICLR)*, 2021.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. EXPO: Stable Reinforcement Learning with Expressive Policies. In *International Conference on Learning Representations (ICLR)*, 2026a.

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

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-Flow: Stable and Expressive Reinforcement Learning with Flow-Based Policy. In *International Conference on Machine Learning (ICML)*, 2026.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596, 2018.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean Flows for One-step Generative Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies. *arXiv preprint arXiv:2304.10573*, 2023.

Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In *International Conference on Learning Representations (ICLR)*, 2016.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient Diffusion Policies for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, and Ya-Qin Zhang. PROTO: Iterative policy regularized offline-to-online reinforcement learning. *arXiv preprint arXiv:2305.15669*, 2023.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement Learning with Action Chunking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2016.

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

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)*, 2023.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning. *Science Robotics*, 10(105), 2025. doi: 10.1126/scirobotics.ads5033.

Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *arXiv preprint arXiv:2303.17396*, 2023.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

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

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Mitsuhiko Nakamoto, Yuexiang 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.090.

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

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, Zeynep Temel, Oliver Kroemer, Sergey Levine, Abhishek Gupta, Hongkai Dai, Paarth Shah, and Max Simchowitz. OGPO: Sample Efficient Full-Finetuning of Generative Control Policies. *arXiv preprint arXiv:2605.03065*, 2026.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan

Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a VLA That Learns From Experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. THE COLOSSEUM: A Benchmark for Evaluating Generalization for Robotic Manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.133.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Stéphane Ross and J. Andrew Bagnell. Efficient Reductions for Imitation Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *Proceedings of Machine Learning Research*, pp. 661–668, 2010.

Stéphane Ross, Geoffrey Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, pp. 387–395, 2014.

Mingyang Sun, Pengxiang Ding, Weinan Zhang, and Donglin Wang. Iterative refinement of flow policies in probability space for online reinforcement learning. *arXiv preprint arXiv:2510.15388*, 2025.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive RL finetuning. In *International Conference on Machine Learning (ICML)*, 2026.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

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 (CoRL)*, 2025.

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

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.

Ziqian Wang, Rui Zhang, Yitian Liu, Xingjian Mao, Minqian Wang, and Yao Mu. Q-VGM: Q-Guided Value-Gradient Matching for Offline-to-Online RL of Flow-Matching VLA Policies. *arXiv preprint arXiv:2606.08015*, 2026b.

Maciej Wołczyk, Bartłomiej Cupiał, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. In *International Conference on Machine Learning (ICML)*, 2024.

Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the Generalization Gap in Imitation Learning for Visual Robotic Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3153–3160, 2024. doi: 10.1109/ICRA57147.2024.10611331.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yuxin Chen, Yiheng Li, Hongyang Li, Masayoshi Tomizuka, and Shengbo Eben Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *International Conference on Learning Representations (ICLR)*, 2026.

Lunjun Zhang, Shuo Han, Hanrui Lyu, and Bradly C. Stadie. D2 actor critic: Diffusion actor meets distributional critic. *Transactions on Machine Learning Research (TMLR)*, 2025a.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning Flow Matching Policy with Online Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC Flow: Sample-Efficient Reinforcement Learning of Flow-Based Policies via Velocity-Reparameterized Sequential Modeling. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient Online Reinforcement Learning Fine-Tuning Need Not Retain Offline Data. In *International Conference on Learning Representations (ICLR)*, 2025.

Christos Ziakas, Alessandra Russo, and Avishek Joey Bose. Aligning flow map policies with optimal Q-guidance. *arXiv preprint arXiv:2605.12416*, 2026.

A Analysis Protocol

Task. Section 3 uses `tool_hang` (proficient-human demonstrations, low-dimensional observations) from Robomimic, chosen because the pretrained policy leaves ample room to improve.

Pretrained policy. Every run starts from the same generator θ_0, pretrained to convergence with the MeanFlow objective on the 200 demonstrations. On a fixed probe set of 64 episodes it succeeds in 41 to 47% of episodes across five readings, and in 49% on its own training-time evaluation. Online, the parameters θ of the action decoder are updated and the observation encoder is kept frozen.

Learning rate. Every run of Section 3 uses a constant actor learning rate of 4.5×10^{-5} unless stated otherwise, below the 3×10^{-4} default of TD3 and SAC (Fujimoto et al., 2018; Haarnoja et al., 2018); the critic learning rate is 3×10^{-4}.

Critic and replay. The critic is the ensemble of Section 2.3 with $N = 10$ heads and random subsets of $M = 2$; every batch of 256 transitions draws 128 from the demonstrations and 128 from the online buffer.

Anchor. The anchored runs of Figure 2a add to the actor loss equation 3 the behavior cloning term $\mathcal{R}(\theta; \mathcal{D}) = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{demo}},\,\epsilon} \|f_\theta(s, \epsilon) - a\|^2$, the squared error between the generated and the demonstrated action chunk, at weights $\lambda = 1$ and 3; every other run sets $\lambda = 0$.

Probe bank. The bank $\mathcal{B}$ of equation 4 holds 32 states drawn once from the demonstrations with a fixed seed, each paired with 8 noise draws, 256 rows in all. The norm in equation 4 is the Euclidean norm of the whole action chunk (7 action dimensions over a horizon of 8 on `tool_hang`), and d is the root mean square of these norms over the bank. The probe of Figure 2b, the proposals of Figure 2c, and the functional step bound δ of Section 4 are all measured in this one quantity.

Budget and evaluation. Runs use one million environment steps unless stated otherwise. Two measurements are reported and never mixed. Training curves are evaluations of 64 episodes taken every 20,000 steps; where a single training-curve number is quoted it is the mean of the last eight evaluations at or below the stated budget (512 episodes). Final success rates are measured on the final checkpoint with 100 held-out episodes and fixed reset seeds. Evaluation labels count online steps plus the 10,000 critic warm-up steps. With the bound in place, each evaluation is taken at the first episode boundary at or after its scheduled step, on the committed actor.

Seed-to-seed spread. Four seeds of one configuration (the unanchored direct update at 4.5×10^{-5}) end at 33, 35, 48, and 55% on the held-out evaluation.

Probe. The single-step probe of Figure 2b takes the critic at the end of a successful run (its policy succeeds in 93% of episodes), freezes it, computes the direct-update gradient once at θ_0 on a batch of 256 demonstration observations, and evaluates $\theta_0 - L\,\hat{g}$ for a grid of L on the same 64-episode probe set with the same reset seeds. The functional step on the horizontal axis is the same quantity that the projection of Section 4 bounds. Five independent gradient draws were probed; Figure 2b shows three of them singly and the mean of all five.

Figure 2. Panel (a) shows four seeds of the direct update with RLPD, two seeds at each anchor weight ($\lambda = 1, 3$), two without RLPD, four at the 150-fold smaller learning rate (target rate $\tau = 0.005$), and two at that learning rate with the functional step bound of Section 4 ($\delta = 0.01$, target rate $\tau = 0.05$; this is a method reference, since τ differs; the comparison at matched τ is the bound row of Table 2 and Appendix C; the second run stops at 430k steps, and the shading ends with it); lines are the mean of the evaluations in a centered window of a tenth of the budget (five evaluations at the larger learning rate, seven at the smaller), pooled over seeds, shading is one standard deviation of the same pool, and the gray band spans the five readings of θ_0 above. Panel (b) shows the five gradient draws of the probe, three of them singly and all five in the mean, with one standard deviation across draws. The dashed curve moves θ_0 along three random unit directions in parameter space, each scaled so that its functional step on the probe bank matches the abscissa (within 1%); a random direction needs about a hundred times the parameter displacement of the gradient for the same functional step. Panel (c) shows the functional step proposed in each episode as medians over 50,000-step bins, with lines and shading as in (a), for the runs of (a) without the bound at both learning rates; the gray interval is the phase in which the unanchored runs of (a) have zero success.

B Drift in Action Space

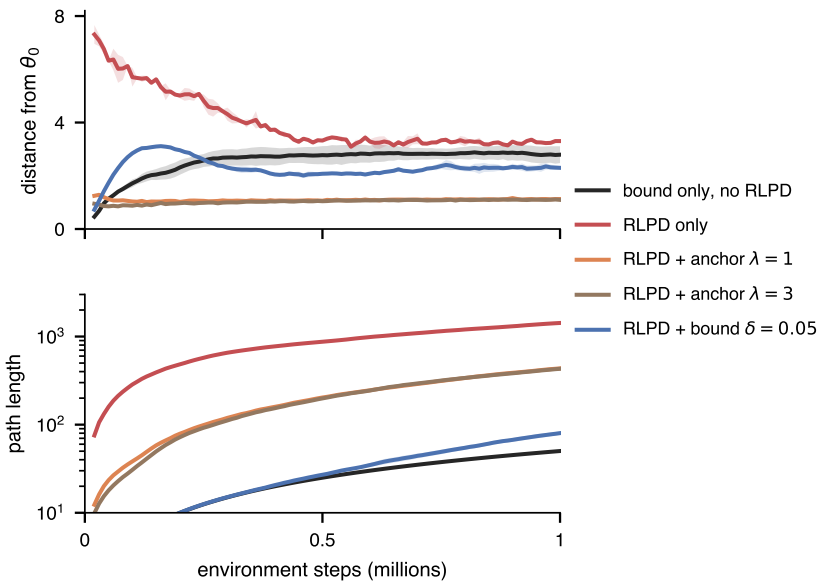


Figure 7: Drift in action space, in the metric that the projection bounds, for the runs of Figure 2a at the learning rate of Section 3 (4.5×10^{-5}): the update without the bound, the anchor at $\lambda = 1$ and 3, the bound at $\delta = 0.05$ (the value of those runs), and the bound without RLPD. Top: net distance of the policy from θ_0. Bottom: cumulative path length, log scale. Lines are means, shading one standard deviation across seeds.

Figure 7 separates what the anchor and the bound control, at the learning rate and bound of Section 3 (4.5×10^{-5}, $\delta = 0.05$). The anchor keeps the policy near θ_0 (a net distance of about 1.1 throughout), the bound does not (2.3, against 3.3 without it); what the bound changes is the path, whose length falls from about 1,400 without the bound to 80 with it, while the anchor lies in between at about 430. The update without the bound first moves far from θ_0 (above 7 within the first episodes) and returns only slowly, which is the collapse and partial recovery of Figure 2a seen from the policy's side.

At the learning rate of the main results. At 3×10^{-7} with $\delta = 0.01$ the projection binds at every episode boundary, so the committed step equals the bound and the drift is set by the number of episodes and the direction of the steps. Adding the anchor of Section 3.2 on top of the bound changes what the optimizer proposes (Table 3): over the first 300k steps, the raw functional step per episode falls from 0.15 without the anchor to 0.08 at $\lambda = 3$ and 0.018 at $\lambda = 150$ on `tool_hang`, and on `kitchen-partial` from 0.064 to 0.008 at $\lambda = 150$, below the bound itself, so that the projection binds in 11% of episodes (2% after 900k steps). The net distance from θ_0 after 300k steps is 1.00 without the anchor against 0.61 at $\lambda = 3$ and 0.73 at $\lambda = 150$ on `tool_hang`, and 0.60 against 0.36 on `kitchen-partial`, where the anchored policy then stays at 0.38 while the unanchored one moves out to 1.78 by the end of the budget.

	λ	episodes	raw step	distance from θ_0	projected
`tool_hang`, first 300k	0	445	0.152	1.00	100%
	3	457	0.077	0.61	100%
	150	322	0.018	0.73	99%
`kitchen-partial`, first 300k	0	524	0.064	0.60	100%
	150	501	0.008	0.36	11%
`kitchen-partial`, after 900k	0	1732	0.023	1.78	100%
	150	901	0.007	0.38	2%

Table 3: The bound with and without the anchor at the main line (3×10^{-7}, $\delta = 0.01$, $\tau = 0.05$), one run each: episodes in the window, median raw functional step proposed per episode, net distance from θ_0 in action space at the end of the window, and the fraction of episodes in which the projection binds. The first-300k rows cover the same 300k environment steps, over which the arms complete different numbers of episodes; the last two rows cover the run from 900k steps on.

C Training Curves for the Ablations

Figure 8 shows the runs behind the bound and demonstration rows of Table 2, and Figure 9 the bound-size and learning-rate rows, at the learning rates of the main results.

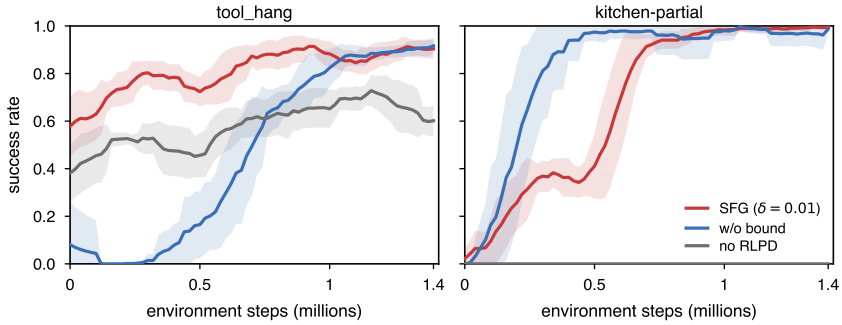


Figure 8: Curves of the bound and demonstration arms of Table 2, smoothing and shading as in Figure 3; on `kitchen-partial`, success is all four subtasks completed, as in that table.

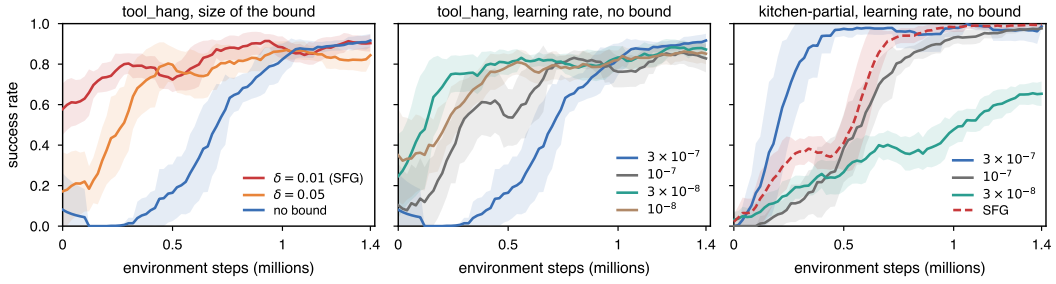


Figure 9: The bound-size and learning-rate rows of Table 2: (a) `tool_hang` with $\delta = 0.01$, $\delta = 0.05$, and no bound; (b) `tool_hang` and (c) `kitchen-partial` at smaller learning rates without the bound, with the SFG run of (c) dashed for reference. Smoothing and shading as in Figure 3. The $\delta = 0.005$ rows, the `kitchen-partial` $\delta = 0.05$ row, and the 10^{-8} row on `kitchen-partial` are reported in the table only.

Details of Table 2. Without the bound, success on `tool_hang` is zero over the first 0.3M steps and the first successful evaluation comes at 400k steps. On `kitchen-partial` the run with the bound climbs later than the one without (0.21 against 0.36 over the first 0.3M steps) and ends two points higher. A tighter bound ($\delta = 0.005$) keeps more of the early success on `tool_hang` (0.69) but ends lower on both tasks (0.83 and 0.85); a looser one ($\delta = 0.05$) is the fastest on

kitchen-partial (0.66 early) but already loses a third of the pretrained success on tool_hang (0.37). Among the smaller learning rates without the bound, the best early success on tool_hang is 0.56 at 3×10^{-8}, against 0.65 with the bound, and on kitchen-partial training stalls at 0.66 with 3×10^{-8} and does not start at 10^{-8}. Removing the demonstration half of every batch with the bound in place leaves tool_hang at 0.65 and kitchen-partial at zero throughout; removing RLPD and the bound together gives zero on both tasks. A further run, not in Table 2, adds the behavior cloning term of Section 3.2 to the update with the bound at $\lambda = 150$, the weight of that section raised by the ratio of the two learning rates; it ends at 0.20 and 0.24 on tool_hang and 0.01 and 0.13 on kitchen-partial, against 0.65 and 0.90 and 0.21 and 0.99 with the bound alone.

D THE REMAINING RUNS ON TRANSPORT

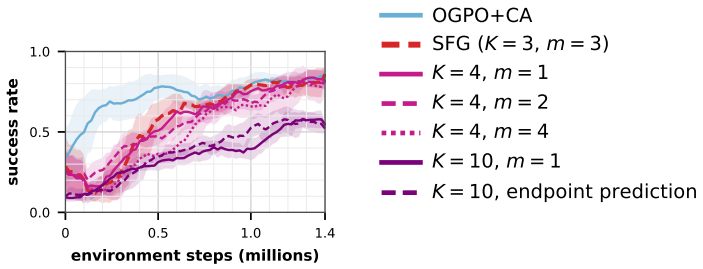


Figure 10: The other multi-step configurations of SFG on transport: the flow-matching policy of OGPO decoded with $K = 4$ Euler steps and trained through the last one, two, or all four of them (Algorithm 2), and decoded with $K = 10$ steps and trained through the last step or by the endpoint estimator (Algorithm 3); the $K = 3$ run of Figure 4 and OGPO+CA for reference. Four seeds each, $\delta = 0.01$, conventions as in Figure 3 with θ_0 at step 0.

Figure 4 shows the two versions of SFG and every baseline on transport; the remaining multi-step configurations are collected here. Among the multi-step configurations (Figure 10), the runs with three or four Euler steps share one shape, a drop from θ_0 to about 10% in the first 200k steps, a return to 50% by 500k, and a slow climb to about 80% at 1.4M; how many of the steps the gradient passes through changes the endpoint by less than the spread of a single 64-episode evaluation (81 to 83%). The ten-step policy starts at 13% and reaches 57% with the gradient through the last step and with the endpoint estimator alike. The $K = 3$ policy without the bound is at zero from its first evaluation on (not drawn). Of the baselines in Figure 4, DFP and MVP, the two one-step generators anchored to the data, end at 13 and 5%, and DSRL and QC-FQL stay below 5% throughout, as does the one-step version of SFG, whose run was stopped at 950k steps after nineteen zero evaluations.

E MEANFLOW PRETRAINING DETAILS

The average velocity of the flow of Section 2.1 over an interval is $u(z_t, r, t) = \frac{1}{t-r} \int_r^t v(z_\tau, \tau)\, \mathrm{d}\tau$, so that $z_r = z_t - (t - r)\, u(z_t, r, t)$ holds exactly for every $r < t$. Differentiating this definition in t gives the MeanFlow identity

$$u(z_t, r, t) = v(z_t, t) - (t - r)\, \frac{\mathrm{d}}{\mathrm{d}t}\, u(z_t, r, t), \qquad \frac{\mathrm{d}}{\mathrm{d}t}\, u = v\, \partial_z u + \partial_t u, \tag{6}$$

where the total derivative is taken along the flow with r held fixed. The network $u_\theta(z_t, r, t \mid s)$ is trained to satisfy this identity by regressing onto its right-hand side, with the conditional velocity $\epsilon - a$ in place of v and the total derivative evaluated by a Jacobian-vector product whose gradient is stopped,

$$\mathcal{L}_{\mathrm{MF}}(\theta) = \mathbb{E}_{(s,a),\, \epsilon,\, (r,t)} \left\| u_\theta(z_t, r, t \mid s) - \mathrm{sg}\left[(\epsilon - a) - (t - r)\, \frac{\mathrm{d}}{\mathrm{d}t}\, u_\theta(z_t, r, t \mid s) \right] \right\|^2, \tag{7}$$

with $r \leq t$; at $r = t$ the target reduces to the flow matching target of equation 1. We use this objective in the form of Geng et al. (2025), with three additions that we state for completeness. *Time sampling.* The pair (r, t) is drawn with $r = t$ on half of the samples and $r < t$ on the other half. *Guidance.* The conditional velocity $\epsilon - a$ in the target is replaced by the guided velocity $\tilde{v} = w\,(\epsilon - a) + (1 - w)\,u_\theta(z_t, t, t \mid s)$ with $w = 2$, which folds classifier-free guidance into training so that inference still needs a single evaluation. *Loss weighting.* The squared error is weighted adaptively by $1/(\|\cdot\|^2 + c)^p$ with $p = 0.5$ and $c = 10^{-3}$, computed without gradient. The latent is $\epsilon \sim \mathcal{N}(0, I)$ of the shape of an action chunk. Observations and actions are normalized to $[-1, 1]$ per dimension. Training uses Adam with learning rate 10^{-4}, no weight decay, batch size 128, gradient-norm clipping at 1.0, and an exponential moving average of the weights with decay 0.999; the averaged weights are the policy that is evaluated, saved, and fine-tuned. Checkpoints are screened on rollouts, never on the training loss, which does not track success (Appendix A). The decoder is an MLP of five layers of 512 units on `tool_hang` and four on `square` and the kitchen tasks (Table 4). The checkpoints fine-tuned in Section 5 are taken after 200,000 gradient steps on every task.

F Implementation Details of SFG

Actor loss. The actor minimizes the negative subsampled minimum of equation 3, divided by the batch mean of $|Q|$ with the gradient stopped, so that the scale of the returns does not enter the effective learning rate. Its states are those of the same mixed batch as the critic's (Section 2.3), so demonstration states enter the loss while demonstration actions do not. The generator output is clipped to $\mathcal{A} = [-1, 1]^d$ once, inside the decode, and everything downstream uses the clipped action: execution, the critic's training and evaluation, the target action of equation 2, and the probe bank on which the functional step equation 4 is measured. The actor loss uses the exact gradient of the clip, so that a saturated coordinate receives no gradient; a straight-through estimator through the clip was worse. The observation encoder of the pretrained generator is frozen; only the action decoder is updated.

Critic. The critic is the ensemble of Section 2.3 with $N = 10$ layer-normalized heads. For every update a subset of $M = 2$ heads is drawn without replacement for the target of equation 2 and, independently, another for the actor loss. The action in the target is generated by the current (proposal) actor with fresh noise. Target networks are updated by Polyak averaging with rate $\tau = 0.05$ (0.005 in the runs of Section 3); the critic learning rate is 3×10^{-4}, and every batch of 256 transitions draws 128 from the demonstrations and 128 from the online buffer.

Warm-up. Before the first actor update, $W = 10{,}000$ environment steps are collected with θ_0 (the last incomplete episode is discarded), and the critic is then trained alone for W updates on the same mixed batches as in training; evaluation labels count these steps.

Chunks and updates. The policy emits a chunk of H actions every H environment steps and executes it open loop. Transitions are stored per environment step and enter the online buffer when the episode ends; a batch samples H-step segments and forms the chunk transition $(s_t, a_t, r_t^H, s_{t+H})$ of equation 2. The critic and the actor are updated once per environment step, after that step's action has been executed.

Projection. The functional step is measured on the probe bank of Appendix A. The episode is collected by the proposal actor $\tilde{\theta}$; at the episode boundary, if $d(\theta, \tilde{\theta}) > \delta$, the fraction α is found by a proportional scalar iteration, $\alpha \leftarrow \alpha\,\delta / d(\theta, \theta + \alpha(\tilde{\theta} - \theta))$ from $\alpha = \delta / d(\theta, \tilde{\theta})$, limited to 1 and accepted once the committed step is within 5% of δ on either side, so that a committed step may exceed δ by up to 5%; if eight iterations do not reach the tolerance, the best iterate is committed (96% of projected episodes converge after one evaluation and 4% after two, and the fallback fired once in 123,760 episodes) We use an actor learning rate of 3×10^{-7} and $\delta = 0.01$ on every task.

Hyperparameters. Table 4 lists the settings of every run of Section 5; the runs of Section 3 differ only where Appendix A says so.

Setting	Value
Actor learning rate	3×10^{-7}, Adam, constant; gradient norm clipped at 1
Functional step bound δ	0.01 on every task
Probe bank	32 demonstration states $\times$ 8 noise draws, fixed seed
Projection	proportional iteration, tolerance $\pm 5\%$ of δ, at most 8 iterations; optimizer state kept
Actor loss	subsampled minimum over $M = 2$ critic heads, divided by the batch mean of $\lvert Q \rvert$ (gradient stopped)
Critic	$N = 10$ heads with layer normalization, $M = 2$ for the target and for the actor loss
Critic learning rate	3×10^{-4}, Adam, weight decay 10^{-5}; target rate $\tau = 0.05$
Discount γ	0.99 (`square`, kitchen); 0.999 (`tool_hang`, `transport`)
Chunk length H	4 (`square`, kitchen); 8 (`tool_hang`, `transport`)
Networks	MLPs of 4×512 (`square`, kitchen) or 5×512 (`tool_hang`, `transport`), actor and critic alike
Batch	256 transitions, 128 from the demonstrations and 128 from the online buffer; actor batch 128
Replay buffer	2×10^6 transitions; demonstrations kept throughout
Warm-up	$W = 10{,}000$ steps with θ_0, then W critic updates
Updates	one critic and one actor update per environment step
Budget and evaluation	1.4M online steps; every 20k steps (10k on `square`), 64 episodes
Generator	one-step MeanFlow decode; observation encoder frozen
Multi-step version	flow-matching policy with $K = 3$ (`transport`) or 10 Euler steps, 4 updates per step, 50k warm-up, evaluation every 40k steps

Table 4: Hyperparameters of SFG. Values shared across tasks are stated once; the per-task values follow the pretraining configuration of each benchmark.

G SFG WITH A MULTI-STEP SAMPLER

When the pretrained policy is a flow-matching velocity field v_θ (Section 2.1) rather than a MeanFlow generator, an action is the endpoint of K Euler steps from $z_1 = \epsilon$ to $t = 0$, and f_θ is a composition of K evaluations of v_θ. Algorithm 1 is unchanged: the decode $f_{\tilde{\theta}}$ at lines 5 and 8, the actor loss at line 9, and the probe bank behind $d(\cdot, \cdot)$ all call the same K-step sampler. What is left open is how the critic's gradient is taken through the chain at line 9. We use two rules, given as Algorithms 2 and 3. Both leave the forward pass untouched: execution, the TD target, the probe bank, and evaluation all use the full K-step endpoint, so runs that differ only in the gradient rule execute the same policy at the same parameters and differ only in the direction of the update. Here sg is the stop-gradient (the identity in the forward pass) and clip the action clipping to $\mathcal{A} = [-1, 1]^d$ of Section 2.1; only the final action is clipped, not the intermediate states.

Truncated backpropagation. Algorithm 2 backpropagates through the last m of the K steps and detaches the state entering them. Since the velocity evaluated at a detached step reaches the action only through that state, the first $K - m$ evaluations of v_θ receive no gradient at all; with $m = K$ the gradient passes through the whole decoder, with $m = 1$ through its last evaluation only. The backward pass stores the activations of m chained network evaluations.

Endpoint prediction. Algorithm 3 takes no gradient through the chain. From each intermediate state z at time t_k, $k \geq k_0$, it predicts the endpoint in a single Euler step of length t_k, $\hat{a}_k = \mathrm{clip}(z - t_k\, v_\theta(z, t_k \mid s))$ with z detached, and the actor loss is the mean over these predictions of the normalized critic value, each differentiable through one network evaluation; this is the action approximation of EDP (Kang et al., 2023) applied to a flow sampler. The executed action is still the K-step endpoint; the predictions exist only to carry the gradient, and the critic is evaluated on all of them in one call on a $(K - k_0)$-fold batch. With $k_0 = K - 1$ there is a single prediction, $\hat{a}_{K-1} = \mathrm{clip}(z - \Delta\, v_\theta)$, which is the executed action itself, and the rule coincides with Algorithm 2 at $m = 1$; the two rules differ only in the predictions taken from earlier steps. The normalization is per prediction, because the step of length t_k scales the gradient of the earliest prediction well above that of the last one (a factor of about five between $k = K/2$ and $k = K - 1$ at $K = 10$), so that an unnormalized mean would be dominated by the noisiest prediction.

Algorithm 2 K-step decode with truncated backpropagation (replaces $f_{\tilde{\theta}}$ in Algorithm 1)

Require: velocity field v_θ; state s; noise ϵ; sampler steps K; backpropagated steps $m \in \{1, \dots, K\}$

$z \leftarrow \epsilon$; $\quad \Delta \leftarrow 1/K$
for $k = 0, \dots, K-1$ **do**
  $t_k \leftarrow 1 - k/K$
  $z \leftarrow z - \Delta\, v_\theta(z, t_k \mid s)$ ▷ one Euler step toward $t = 0$
  if $k < K - m$ **then**
    $z \leftarrow \mathrm{sg}(z)$ ▷ the gradient stops here; the value of z is unchanged
  end if
end for
return $a \leftarrow \mathrm{clip}(z)$ ▷ executed, scored by the critic, and measured on the probe bank

Algorithm 3 Endpoint-prediction gradient (replaces line 9 of Algorithm 1)

Require: velocity field v_θ; state s; noise ϵ; sampler steps K; first predicted step k_0 (default $K/2$)

$z \leftarrow \epsilon$; $\quad \Delta \leftarrow 1/K$; $\quad \mathcal{P} \leftarrow \emptyset$
for $k = 0, \dots, K-1$ **do**
  $t_k \leftarrow 1 - k/K$; $\quad v \leftarrow v_\theta\big(\mathrm{sg}(z), t_k \mid s\big)$ ▷ differentiable in θ, not through the chain
  if $k \geq k_0$ **then**
    $\hat{a}_k \leftarrow \mathrm{clip}\big(\mathrm{sg}(z) - t_k\, v\big)$; $\quad$ add $\hat{a}_k$ to $\mathcal{P}$ ▷ one step of length t_k predicts the endpoint
  end if
  $z \leftarrow \mathrm{sg}(z) - \Delta\, \mathrm{sg}(v)$ ▷ the chain is advanced without gradient
end for
$a \leftarrow \mathrm{clip}(z)$ ▷ executed action, TD target, probe bank: the same value as Algorithm 2
update $\tilde{\theta}$ by one optimizer step on $-\frac{1}{|\mathcal{P}|} \sum_{\hat{a}_k \in \mathcal{P}} \mathbb{E}_{s,\epsilon}\big[\min_{i \in \mathcal{M}'} Q_{\phi_i}(s, \hat{a}_k)\big] \big/ \mathrm{sg}\, \mathbb{E}\, |Q(s, \hat{a}_k)|$

Setup. The pretrained policies are the flow-matching policies of the OGPO codebase, trained by behavior cloning and decoded with $K = 10$ Euler steps ($K \in \{3, 4\}$ in Appendix D); on `tool_hang` this policy succeeds in 25% of episodes, on `transport` in 13%. Runs follow the recipe of Appendix F, with these settings: $\delta = 0.01$ and $\tau = 0.05$, an actor learning rate of 3×10^{-7} on every task, a warm-up of 50,000 steps followed by 400,000 critic updates, four critic and four actor updates per environment step, and 1.4M online steps; evaluations of 64 episodes are taken every 40,000 steps (35 per run). Every cell of Table 5 is the mean over four seeds.

Projection. The proportional iteration of Appendix F assumes that the functional step is close to linear in α; through K applications of the actor it is not, and at $K = 10$ the iteration exhausts its budget at every episode boundary and would commit about a third of δ. The runs of this appendix therefore fall back to a bisection in $\log \alpha$ on $[10^{-12}, 1]$ with 24 iterations whenever the iteration misses its tolerance; the solutions lie near $\alpha \approx 10^{-7}$, where a bisection in α itself would spend its whole budget reaching that magnitude. The projection binds at every episode boundary in every run. The fallback is never reached on the kitchen runs, whose committed step is within 1% of δ at the median (residual 0.011 to 0.013), and it is reached in 30 to 63% of the episodes on `tool_hang`; on the last-step arm ($m = 1$) there, the solver misses its tolerance in half of the episodes and, after 1M steps, commits a step of exactly zero in one episode of five (7% of all episodes), so that run is the one whose update was throttled by the solver rather than by δ. The $K = 3$ and $K = 4$ runs of Appendix D commit 0.0101 at the median as well, with the fallback in at most 5% of episodes. Without the functional step bound, the K-step policy on `transport` is at zero from its first evaluation on (Appendix D).

	SFG, Table 1	ten-step flow-matching policy, by gradient rule			
	(one-step)	m=1	m=5	m=10	endpoint prediction
tool_hang	0.91	0.66	0.82	0.76	0.53
kitchen-partial	1.00	**0.93**	0.70	0.68	0.92
transport	0.81†	**0.57**	–	–	**0.57**
square	0.98	0.99	–	–	–
kitchen-mixed	1.00	0.87	–	–	–
kitchen-complete	1.00	0.98	–	–	–

Table 5: SFG on a ten-step flow-matching policy, by the rule that carries the critic's gradient through the sampler: backpropagation through the last m of the ten Euler steps (Algorithm 2) or endpoint prediction from the last five steps, which takes no gradient through the chain (Algorithm 3). Task completion score at the end of the budget under the convention of Table 1: mean of the evaluations at 1.2M steps and later, kitchen as the fraction of subtasks completed; four seeds per cell. The reference column is the SFG row of Table 1 (†transport: the three-step policy of Figure 4). The same runs on the all-four-subtasks metric are quoted in the text; no ordering is read from the tool_hang row, two of whose runs hit the solver fallback.

Results. On tool_hang the three depths end at 66, 82, and 76% for $m = 1$, 5, and 10; the last-step and whole-chain runs are the two whose committed steps the projection solver throttled (previous paragraph), so we read no ordering of m from this task. Endpoint prediction ends at 53% on tool_hang, at 0.92 of the subtasks on kitchen-partial (0.93 for the last-step rule; 74% against 73% on all four), and at 57% on transport, the same as the last-step rule there. The last-step rule at $K = 10$ also reaches 99% on square, 0.98 of the subtasks on kitchen-complete, and 0.87 on kitchen-mixed (96% and 55% of episodes completing all four). On transport the last-step rule reaches 57% at $K = 10$ against 83% at $K = 4$ (Appendix D); the pretrained policy itself succeeds in 52 to 54% of episodes when decoded with two or three steps and in 13% at ten.

Details behind Section 5.3. Decoding the MeanFlow network of the main results with ten steps and backpropagating through all of them, the update on tool_hang still trains without the bound: it leaves zero later than the one-step run, at about 600k steps against 350k, and ends at 78% against 91%, a gap inside the seed-to-seed spread of Appendix A (four seeds each). On the flow-matching policy decoded with ten steps (Figure 5, Table 5), backpropagating through all ten steps reaches 76% on tool_hang and 68% of the subtasks on kitchen-partial, and no depth reaches the one-step generator on either task. With the gradient through the last step, the ten-step policy matches the one-step generator on square and kitchen-complete and falls short where its pretrained policy is weaker, succeeding in 25% of episodes on tool_hang and 13% on transport against 52 and 54% for the policies of Table 1.

H Baselines and Related Methods: What Is Trained and What Holds the Policy

Table 6 places the baselines of Section 5.1 on the two axes of this paper: whether the policy's own weights are trained, and what holds the policy near where it started. Table 8 does the same for the other methods of Section 6, adding how the critic's signal reaches the policy.

	square	tool_hang	k-mixed	k-partial	k-complete	Mean
Task completion score at 1.4M steps (mean of the evaluations in the last 200k)						
DSRL	0.76	0.01	0.45	0.57	0.52	0.46
EXPO	0.83	0.24	0.52	0.77	0.05	0.48
QC-FQL	0.39	0.00	0.96	0.92	0.85	0.63
DFP	0.97	0.42	0.95	0.98	0.92	0.85
MVP	**0.98**	0.67	0.99	0.97	**1.00**	0.92
OGPO+CA	0.84	0.90	0.87	0.87	0.99	0.90
SFG (ours)	**0.98**	**0.91**	**1.00**	**1.00**	**1.00**	**0.98**
Environment steps (M) at which the smoothed score first reaches 0.8						
DSRL	–	–	–	–	–	1.40
EXPO	1.24	–	–	–	–	1.37
QC-FQL	–	–	**0.22**	0.38	0.60	0.80
DFP	0.36	–	1.14	1.08	1.22	1.04
MVP	0.28	–	0.78	1.04	0.56	0.81
OGPO+CA	1.04	0.61	0.78	0.78	**0.01**	0.64
SFG (ours)	**0.18**	**0.28**	0.38	**0.26**	0.28	**0.28**

Table 7: Per-task values behind Table 1: task completion score at the end of the 1.4M-step budget (success rate on Robomimic, fraction of the four subtasks completed on kitchen), and the number of environment steps at which the smoothed score first reaches 0.8 ("–": not within the budget, counted as 1.4M in the mean). Each cell is the mean over four seeds. Best value per column in bold.

Method	Trains the policy's weights	Anchor
DSRL	no (frozen); a policy over its noise	none; the policy itself stays frozen
EXPO	no (frozen); a residual edit policy	the frozen policy, bounded edits
QC-FQL	a one-step actor beside the flow policy	distillation to the flow policy fit to the data
MVP	yes, one-step generator	flow-matching term on the data
DFP	yes, one-step generator	behavior cloning term on the data
OGPO+CA	yes, full policy through the chain	PPO clip to the previous policy; imitation of its own successful episodes
SFG (ours)	yes, full decoder	none; a bound on the functional step of each episode

Table 6: What each method trains and what keeps its policy near where it started. SFG uses no actor anchor toward the demonstrations or the pretrained policy; it controls the change relative to the previous committed actor through projection.

I Relation to Trust-Region and Penalty Methods

The functional step projection is a trust region on the generated actions: once per episode it bounds the root-mean-square change of the actions that a deterministic generator produces on a fixed bank of state–noise pairs equation 4, measured relative to the previous committed actor, so that the reference advances during fine-tuning. It is enforced by a projection at the episode boundary rather than by a penalty or a clipped ratio inside the loss: within the episode the critic's gradient is followed unmodified, and at the boundary the accumulated update is rescaled to the bound, with no coefficient to balance against the objective. TRPO and PPO bound a divergence between consecutive policy distributions (Schulman et al., 2015; 2017), as do SWFP and MPO (Sun et al., 2025; Abdolmaleki et al., 2018); a one-step generator has no tractable density, and the change of its outputs on a fixed probe bank is the analog that a forward pass can measure. The value of the bound is chosen empirically (Section 4.1).

Penalties toward the data or the pretrained policy, including the trust regions that FMQ and TRQAM place around the pretrained flow policy (Ziakas et al., 2026; Dong et al., 2026b) and the proximity terms of some flow-policy methods (Zhang et al., 2026), are anchors in the sense of Section 2.2: they constrain where the policy may end. TD3-C and PROTO (Luo et al., 2023; Li et al., 2023) bound the step from the previous policy for Gaussian actors and are the closer relatives, with the divergence replaced here by the functional step. Section 3 motivates the bound by the loss of pre-

Method	Trains the policy's weights	Critic reaches the policy by	Anchor or constraint (reference)
Diffusion-QL	yes, through the chain	direct Q-gradient through the chain	denoising BC loss on the data (data)
IDQL	BC fit only	reweighting its samples	the policy is the BC fit (data)
FQL	a one-step actor beside the BC flow	direct Q-gradient into the one-step actor	distillation to the BC flow (data)
PA-RL	yes	supervised fit to critic-optimized samples	the samples start from its own policy (own samples)
DPPO	last denoising steps	on-policy gradient over the denoising MDP	PPO clip (previous iterate)
ReinFlow	yes	on-policy gradient over the noise-injected chain	PPO clip (previous iterate)
FPO	yes	on-policy; flow matching loss ratio in the PPO clip	PPO clip (previous iterate)
SAC Flow	yes, through the chain	direct Q-gradient through the reparameterized chain	$\beta \\|a_\theta - a\\|^2$ to replay actions when fine-tuning (data)
Q-Flow	yes	velocity matching to a flow matching target plus the value gradient	the flow matching target (data); offline
Q-VGM	yes	velocity matching to a normalized, gated Q-gradient step from its own endpoint	no separate term; the step is bounded (own endpoint)
TRQAM	yes	adjoint matching	adaptive path-space KL (pretrained policy)
FMQ	yes	regression onto a Q-guided target	radius η around the reference policy (pretrained policy)
OFQL	yes, one-step	direct Q-gradient	MeanFlow loss on the data (data); offline
MeanFlowQL	yes, one-step	direct Q-gradient	MeanFlow loss on the data (data)
DICE-RL	no (frozen); a residual	direct Q-gradient into the residual	selective penalty $\\|s_\theta\\|^2$ on the residual (pretrained policy)
SWFP	appended flow blocks	soft-Q energy with entropy	W_2 penalty (previous iterate)
TD3-C	yes (deterministic)	direct Q-gradient	ℓ_2 bound to the target policy (previous iterate)
PROTO	yes (Gaussian)	direct Q-gradient	KL penalty (previous iterate)
SFG (ours)	yes, full decoder	direct Q-gradient	none; a bound on the functional step (previous committed actor)

Table 8: The methods cited in Section 6 on the axes of Table 6, with the path by which the critic's signal reaches the policy. Each row follows the method's own paper; the reference of the anchor or constraint is in parentheses. The methods of Table 6 are not repeated. Among the methods that train the pretrained policy's own weights, only Q-VGM and SFG add no term toward the data or the pretrained policy; Q-VGM bounds a per-update step of a surrogate target, SFG bounds the per-episode change of the committed actor.

trained competence within the first updates; whether it also limits the exploitation of critic errors is not separated by our experiments.